

Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

**REAL-TIME
PROGRAMMING**

RECORDING

TRANSLATING

PERFORMANCE



LANGUAGES:
C, Pascal,
Forth

Turbo Languages are Super!



like a rocket ...

“Turbo Basic compiles faster than anything I have seen.” —Ethan Winer, *PC Magazine*

1987 *Programmer's Journal*

Powerful features include:

- Producing EXE files
- Separate compilation
- Built-in project management
- Graph unit including support for IBM CGA, EGA, VGA, and 3270, Hercules and ATT 6300
- Online, context-sensitive help

*Run on an 8 MHz IBM PC AT.

Add expertise:

The Turbo Pascal Toolboxes

Start with Turbo Pascal Tutor for just \$69.95 and add the others as your interests and expertise grow:

- Database Toolbox
- Editor Toolbox
- Graphix Toolbox
- Numerical Methods Toolbox
- GameWorks

Toolboxes require Turbo Pascal 4.0

Just \$99.95 each

“Each new Turbo Pascal 4.0 Toolbox is a virtual treasure of programming methods and tips.

—Giovanni Perrone, *PC Week*”



Turbo Basic® is the lightning-fast Basic compiler with a total development environment that puts you in full control.

Even novices can

write professional programs with Turbo Basic's full-screen windowed editor, pull-down menus, and trace debugging system. You also get a long list of innovative Borland features like binary disk files, true recursion, and increased compilation control. Plus the ability to create programs as large as your system's memory can hold—not just a cramped 64K.

The choice is basic: Turbo Basic!

Just \$99.95!

“Turbo Basic, simply put, is an incredibly good product . . . Not only is this the most advanced BASIC ever, but Borland has lived up to its Turbo tradition.

—William Zachmann, *Computerworld*”

Add another Basic advantage: The Turbo Basic Toolboxes

- The Database Toolbox
- The Editor Toolbox

Toolboxes require Turbo Basic 1.1

Just \$99.95 each.

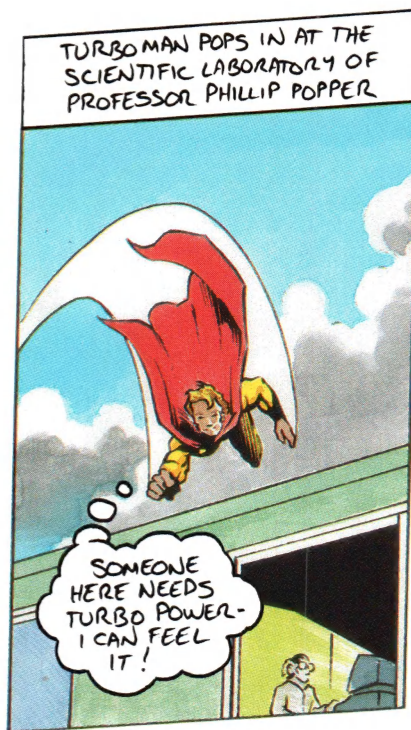
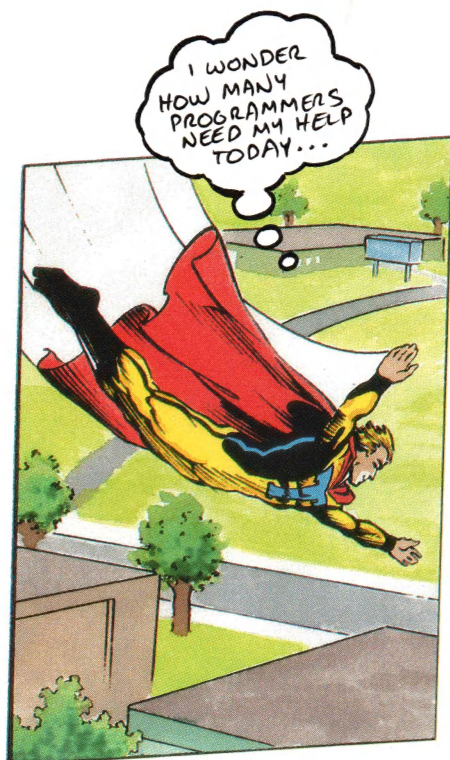
System Requirements For the IBM PS/2™ and the IBM® family of personal computers and all 100% compatibles. PC-DOS (MS-DOS) 2.0 or later. 384K RAM. 640K to compile toolboxes.

Compare the BASIC differences

	<i>Turbo Basic 1.1</i>	QuickBASIC 4.0 Compiler	QuickBASIC 4.0 Interpreter
Compile & Link to stand-alone EXE	3 sec.	7 sec.	—
Size of .EXE	28387	25980	—
Execution time w/80287	0.16 sec.	16.5 sec.	21.5 sec.
Execution time w/o 80287	0.16 sec.	286.3 sec.	292.3 sec.

The Elkins Optimization Benchmark program from March 1988 issue of Computer Language was used. The Program was run on an IBM PS/2 Model 60 with 80287. The benchmark tests compiler's ability to optimize loop-invariant code, unused code, expression and conditional evaluation.

The Critics Agree: Borland's



"Turbo C ... will stun you with in-RAM compilations that operate at warp speed."

—Richard Hale Shaw, *PC Magazine*



Turbo C's® sleek compiler is so fast and powerful, we used it to write our equation solver, Eureka.™ Even better, all that muscle is wrapped in a smooth,

integrated environment with everything you need to make writing, editing and compiling your programs a snap:

- Compiles 10,000 lines per minute*
- Online, context-sensitive help
- ANSI compatible
- Six memory models—tiny to huge
- 450 library functions
- Utilities: Librarian, Make, GREP
- Source code for MicroCalc spreadsheet
- Command-line version of the compiler
- Inline assembly that lets you mix C and assembler

- Professional-quality graphics library supporting VGA, CGA, EGA, Hercules, and IBM 8514
- Interfaces with Turbo Pascal 4.0 and Turbo Prolog

Just \$99.95

“A lightning fast, fully featured C compiler suitable for almost everything ... Borland's Turbo C compiler is flexible, fast and friendly.

—Peter Feldman, *PC Week* ”

Heap Sort

	<i>Turbo C 1.5</i>	Microsoft C 5.0
Compile time	4.7 sec.	16.3 sec.
Compile & link time	7.4 sec.	19.5 sec.
Execute time	10.5 sec.	15.5 sec.
Object code size	1119	1313
Execution size	6392	7891

Sort benchmark run on an 8 MHz IBM AT using Turbo C version 1.5 and the Turbo Linker version 1.1; Microsoft C version 5.0 and the MS overlay linker version 3.61.

“Turbo Pascal 4.0 flies 4.0 is ballistic!” —Tom Swan,



Almost from its introduction, Turbo Pascal® has been the world-wide Pascal standard. It's fast. It's flexible. It's affordable. And it

gives you full control.

Compile more than 27,000 lines of code per minute*. And work in a complete, integrated programming environment with pull-down menus and a full-featured editor.

You don't have to swap code in and out to beat the 64K barrier; it's designed for large programs. Break your code into convenient modules and work with them swiftly and separately. If there's an error in one, you can see it and fix it.

System Requirements For the IBM PS/2™ and the IBM® family of personal computers and all 100% compatibles. PC-DOS (MS-DOS) 2.0 or later. 384K RAM.

† Customer satisfaction is our main concern: If within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.

All Borland products are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks of their respective holders. Copyright ©1988 Borland International, Inc. BI 1230

System Requirements For the IBM PS/2™ and the IBM® family of personal computers and all 100% compatibles. PC-DOS (MS-DOS) 2.0 or later. 384K RAM.

BORLAND PRESENTS

THE ADVENTURES OF TURBOMAN

"Fastest and most approachable
implementation of that language"

—Darryl Rubin, *AI Expert*, on Turbo Prolog

"Most powerful
version of Basic ever"

—Ethan Winer, *PC Magazine*, on Turbo Basic

And able to leap
onto "new ground in the
price/performance arena"

—John H. Mayer, *Computer Design*,
on Turbo C

See the technological
excellence of Turbo C,
Turbo Pascal and
Turbo Basic!

Meet Turbo Prolog 2.0:
Artificial Intelligence like
you've never seen it!

BORLAND

TURBOMAN VISITS SILICON MOTORS, SILICON CITY'S LARGEST MANUFACTURING PLANT...

HELP US TURBOMAN! WE'RE TRYING TO DESIGN A MAPPING SYSTEM INTO OUR NEW CARS, AND WE'RE STUCK!

THIS IS A JOB FOR NEW TURBO PROLOG 2.0 - ITS ARTIFICIAL INTELLIGENCE WILL DO ALL THE HARD WORK FOR YOU!



NEW!

Turbo Prolog 2.0: Powerful Artificial Intelligence for your real-world applications!



New Turbo Prolog® 2.0 lets you harness powerful AI techniques. And you don't have to be an expert programmer or artificial intelligence genius!

You get an all-new Prolog compiler that's been optimized to produce smaller and more efficient programs than ever before. An improved full-screen, completely customizable editor with easy pull-down menus. All-new documentation, including a tutorial rich with examples and instructions to take you all the way from basic programming to advanced techniques. Even online help!

More new features!

- An external database system for developing large databases. Supports B+ trees and EMS
- Source code for a fully-featured Prolog interpreter written entirely in Turbo Prolog. Plus step-by-step instructions to adapt it or include it as is in your own applications!
- Support for the Borland Graphics Interface, the same professional-quality graphics in Turbo Pascal, Turbo C, and Quattro
- Improved windowing
- Powerful exception handling and error trapping features
- Full compatibility with Turbo C so the two languages can call each other freely
- Supports multiple internal databases
- High-resolution video support

Just \$149.95!

60-Day Money-back Guarantee †

For the dealer nearest you
Call (800) 543-7543

CIRCLE NO. 95 ON READER SERVICE CARD

Turbo Prolog Toolbox is 6 toolboxes in one!

More than 80 tools and 8,000 lines of source code help you build your own Turbo Prolog applications. Includes toolboxes for menus, screen and report layouts, business graphics, communications, file-transfer capabilities, parser generators, and more!

Toolbox requires Turbo Prolog 2.0

Just \$99.95

“ If I had to pick one single recommendation for people who want to try to keep up with the computer revolution. I'd say, 'Get and learn Turbo Prolog.' ”

—Jerry Pournelle, *Byte*

An affordable, fast, and easy-to-use language.

—Darryl Rubin, *AI Expert* ”

System Requirements for the IBM PS/2™ and the IBM® family of personal computers and all 100% compatibles. PC-DOS (MS-DOS) 2.0 or later. 384K RAM.



Interlocking Pieces: Blaise and Turbo Pascal.

Whether you're a Turbo Pascal expert or a novice, you can benefit from using professional tools to enhance your programs. With Turbo POWER TOOLS PLUS™ and Turbo ASYNCH PLUS™, Blaise Computing offers you all the right pieces to solve your 4.0 development puzzle.

Compiled units (TPU files) are provided so each package is ready to use with Turbo Pascal 4.0. Both POWER TOOLS PLUS and ASYNCH PLUS use units in a clear, consistent and effective way. If you are familiar with units, you will appreciate the organization. If you are just getting started, you will find the approach an illustration of how to construct and use units.

◆ **POWER TOOLS PLUS** is a library of over 180 powerful functions and procedures like fast direct video access, general screen handling including multiple monitors, VGA and EGA 50-line and 43-line text mode, and full keyboard support, including the 101/102-key keyboard. Stackable and removable windows with optional borders, titles and cursor memory provide complete windowing capabilities. Horizontal, vertical, grid and Lotus-style menus can be easily incorporated into your programs using the menu management routines. You can create the same kind of moving pull down menus that Turbo Pascal 4.0 uses.

Control DOS memory allocation. Alter the Turbo Pascal heap size when your program executes. Execute any program from within your program and POWER TOOLS PLUS automatically compresses your heap memory if necessary. You can even force the output of the program into a window!

Write general interrupt service routines for either hardware or software interrupts. Blaise Computing's unique intervention code lets you develop memory resident (TSRs) applications that take full advantage of DOS capabilities. With simple procedure calls, "schedule" a Turbo Pascal procedure to execute either when pressing a "hot key" or at a specified time.

◆ **ASYNCH PLUS** provides the crucial core of hardware interrupts needed to support asynchronous data communications. This package offers simultaneous buffered input and output to both COM ports, and up to four ports on PS/2 systems. Speeds to 19.2K baud, XON/XOFF protocol, hardware handshaking, XMODEM (with CRC) file transfer and modem control are all supported. ASYNCH PLUS provides text file device drivers so you can use standard "Readln" and "Writeln" calls and still exploit interrupt-driven communication.

The underlying functions of ASYNCH PLUS are carefully crafted in assembler and drive the hardware directly. Link these functions directly to your application or install them as memory resident.

Blaise Computing products include all source code that is efficiently crafted, readable and easy to modify. Accompanying each package is an indexed manual describing each procedure and function in detail with example code fragments. Many complete examples and useful utilities are included on the diskettes. The documentation, examples and source code reflect the attention to detail and commitment to technical support that have distinguished Blaise Computing over the years.

Designed explicitly for Turbo Pascal 4.0, Turbo POWER TOOLS PLUS and Turbo ASYNCH PLUS provide reliable, fast, professional routines—the right combination of pieces to put your Turbo Pascal puzzle together. **Complete price is \$129.00 each.**

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

Now, for
Turbo Pascal 4.0!

THE BLAISE M E N U

Turbo POWER SCREEN \$129.00
NEW! General screen management; paint screens; block mode data entry or field-by-field control with instant screen access. Now for Turbo Pascal 4.0, soon for C and BASIC.

Turbo C TOOLS \$129.00
Full spectrum of general service utility functions including: windows; menus; memory resident applications; interrupt service routines; intervention code; and direct video access for fast screen handling. For Turbo C.

C TOOLS PLUS \$129.00
Windows; menus; ISRs; intervention code; screen handling and EGA 43-line text mode support; direct screen access; DOS file handling and more. Specifically designed for Microsoft C 5.0 and QuickC.

ASYNCH MANAGER \$175.00
Full featured interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem control and XMODEM file transfer. For Microsoft C and Turbo C or MS Pascal.

PASCAL TOOLS/TOOLS 2 \$175.00
Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS-Pascal.

KeyPilot \$49.95
"Super-batch" program. Create batch files which can invoke programs and provide input to them; run any program unattended; create demonstration programs; analyze keyboard usage.

EXEC \$95.00
NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

RUNOFF \$49.95
Text formatter for all programmers. Written in Turbo Pascal; flexible printer control; user-defined variables; index generation; and a general macro facility.

TO ORDER CALL TOLL FREE
800-333-8087

TELEX NUMBER - 338139

NOW

THE SEARCH IS OVER! Peter Norton's Online Guides Instant Access Program eliminates most manual searches with a few simple keystrokes. Now, when you order our featured product, we'll send you its Online Database along with Peter Norton's Instant Access Program, absolutely free!

Microsoft and QuickC are registered trademarks of Microsoft Corporation. Turbo Pascal is a registered trademark of Borland International.

ARTICLES

- Real-time** ► **Writing Real-Time Programs Under Unix** **18**
by Bill Cramer
 Porting a real-time process to a Unix environment isn't always an impossible task. In this article, Bill offers some guidelines and C routines to let you either simulate a real-time OS with Unix, or reorganize the program with a message-passing scheme.
- Real-time** ► **Message-Passing Operating Systems** **34**
by Dan Hildebrand
 An overview of operating system design using intertask messaging, with real-world examples from QNX.
- A Simple Decompiler** **50**
by William May
 Bill May returns to *DDJ* with a simple algorithm for recreating source code from tokens. The example source code is in C.

REVIEWS

- EXAMINING ROOM** **124**
coordinated by Ron Copeland
 Products examined from the programmer's perspective. This month: The Peabody online programmer's reference, the C-Index+ data file management library, and T-Debug Plus 4.0.

COLUMNS

- C Pointers** ► **C CHEST** **80**
by Allen Holub
 Allen tackles some of the nastiest bugs around with a set of subroutines for tracking down pointers that muck with `malloc()` and `free()`.
- Forth** ► **THE FORTH COLUMN** **90**
by Martin Tracy
 Martin offers some thoughts on the state of real-time Forth, as well as source code for floating-point extensions.
- Pascal** ► **STRUCTURED PROGRAMMING** **98**
by Kent Porter
 Kent examines the new release of Microsoft's Pascal compiler and compares it to Turbo Pascal 4.0; he also takes a short, painful look at Intel's LIM 4.0 upgrade.
- Transputers** ► **PROGRAMMING PARADIGMS** **110**
by Michael Swaine
 Notes on objectivity at Software Development '88, as well as Prolog, parallelism, and transputers.

FORUM

- EDITORIAL** **6**
by Jonathan Erickson
- RUNNING LIGHT** **8**
by Tyler Sperry
- LETTERS** **12**
by you
- CARTOON** **14**
by Joe Sikoryak
- SWAINE'S FLAMES** **152**
by Michael Swaine

PROGRAMMER'S SERVICES

- OF INTEREST** **140**
 brief product descriptions
- ADVERTISER INDEX** **145**
 Where to go for more information on products.
- PROGRAMMER'S MARKET PLACE** **148**
 classified ads



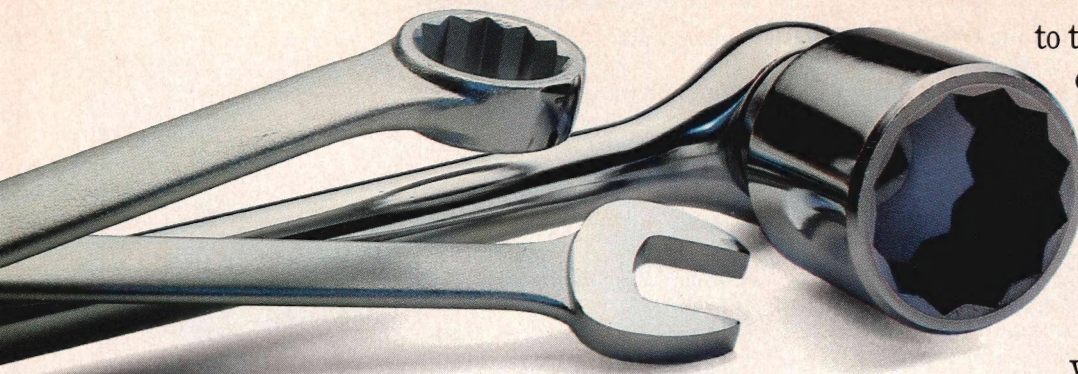
About the Cover

No jokes, please, about code so ugly it could stop a clock. The pocket watch was our second idea: We'd originally planned to do an inner view of a task scheduler for the cover, but it didn't work because we were always getting interrupted.

Next Issue

July will see us covering the frontiers of databases: including some hard information on programming for SQL environments and other goodies.

Buy Our Tools, And We'



Introducing Emerald Bay. The breakthrough database server technology for developing single and multi-user applications. Emerald Bay provides your programs a common data storage and retrieval method which allows data to be transparently shared across multiple and diverse applications.

And when you buy one of our tools for "C", dBASE™ or Lotus® developers, we'll give you the personal engine—free. No royalties to pay, no licenses to sign.

Developed by Wayne Ratliff, the creator of dBASE, Emerald Bay is much more than just another DBMS product, it's an entirely new way to manage data. It's designed to provide an open platform for developing applications in several languages and environments, while Emerald Bay maintains data security, concurrency and integrity.

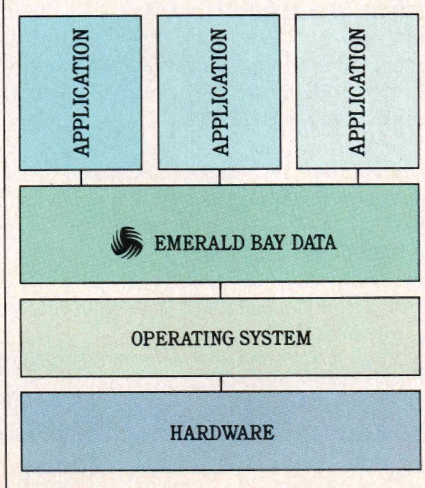
How The Engine Works

Before, data couldn't be readily shared between applications. But with Emerald Bay, PC appli-

cations each share a common data storage and retrieval method. And although the functions of the applications may vary widely, any one application can share another's data transparently; there is no data conversion or translation necessary.

When a PC is an intelligent

Emerald Bay Architecture



workstation on a LAN, the Emerald Bay database server technology controls all data security and integrity, including transaction logging with roll-back. An application simply makes a request, which is sent

to the engine. There, only the essential data is sent back to the workstation. The result is vastly reduced network traffic and faster data access times.

How You Work With The Tools

With the tools we provide, you can easily develop Emerald Bay applications immediately in your familiar development environment.

Emerald Bay technology handles the usually code-intensive management of data, so you can concentrate on what you do best—developing applications.

The *Developers Toolkit for "C"* includes well-documented, easy to use "C" libraries that give you the power to create advanced applications, without the effort usually associated with designing and coding a database "backend".

Eagle is Emerald Bay's sophisticated dBASE-like programming language. As the logical evolution of database language, Eagle introduces advanced features, routines and language components, including a compiler, network commands, user-defined functions in "C" and Assembly and automatic index maintenance.

Summit is an "add-in" database management system for

ll Give You The Engine.

Lotus 1-2-3, which gives you sophisticated data manipulation and analysis commands.

All three of Emerald Bay's development tools come with the Core Components which include Report Writer, Forms Generator, an Import/Export facility and the Database Administrator.

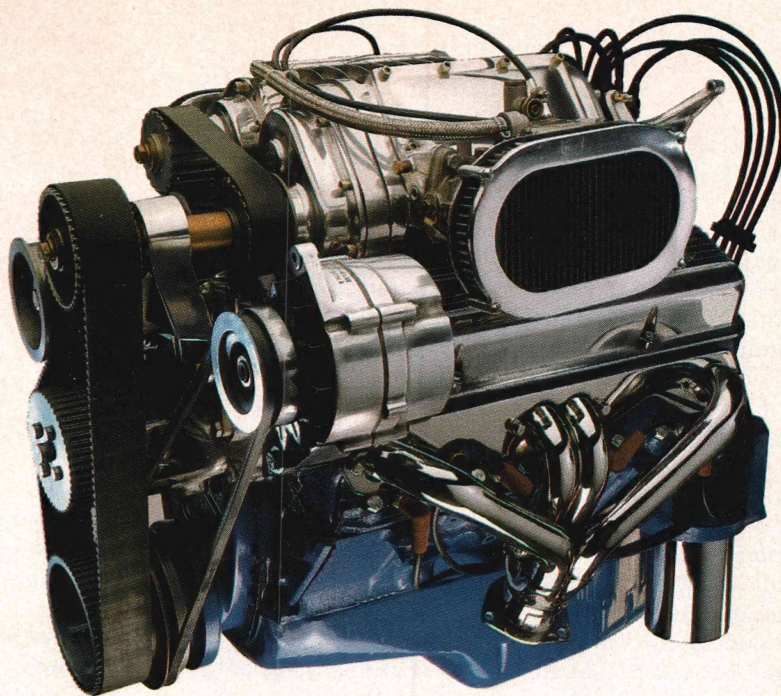
The ***Emerald Bay Database Server*** is the heart of the multi-user Emerald Bay technology. Its client/server architecture is superior to current implementations of LAN/DBMS products, and increases total system throughput, while reducing network traffic and access times.

Finally, while providing a path to other operating systems such as OS/2, Macintosh and UNIX, Emerald Bay is a microcomputer-based technology that optimizes your *current* hardware investment.

How To Find Out More

To get more information about Emerald Bay and for the location of the dealer nearest you, call toll-free 1-800-777-2027.

Emerald Bay. Advanced database server technology. Available *now*.



Emerald Bay Engine Specifications

System Requirements

- MS-DOS 3.1 or greater
- Network database server or Single-user computer: PC XT, AT, PS/2 or 386 compatible, 640K, Hard Disk
- Workstation on LAN: PC, XT, AT, PS/2 or 386 compatible, 640K
- NetBIOS compatible networks supported

Index Storage

- Composite keys supported
- Mixed data type keys allowed
- Keys of up to 100 bytes in length
- Automatic index maintenance
- Ascending and descending keys
- Case independent keys
- Automatic table indexing on record number

Security And Integrity Features

- Access permissions by Read, Write, Delete, Add and Grant
- All five access permissions work on tables and objects
- Read, Write and Grant access permissions operate at field level
- All data other than binary fields can be encrypted
- Transaction logging, with commit and rollback functions
- Full security functions at field and table level
- Optional data encryption at field level

Data Storage

- Max. databases No limit
- Max. tables per database 1000
- Max. fields per table 800
- Max. field width 512 characters
- Max. records per table No limit
- Max. length of records 10,000 bytes (no limit on ext. fields)



 **EMERALD
BAY**

MIGENT™

865 Tahoe Blvd., Call Box 6, Incline Village, NV 89450

Trademarks pending: Emerald Bay, Eagle, Summit (Migent, Inc.)

CIRCLE NO. 164 ON READER SERVICE CARD

EDITORIAL

It's becoming clear that one of the more pressing problems facing software developers will be the ability to find enough qualified programmers to write the software that needs to be written. According to many companies we've talked to, there's a shortage of good programmers right now and the indications are that the situation won't be getting any better.

A couple of recent conversations brought this to mind. In one instance, the president of a major software company mentioned to us that he is getting much of his programming done in Mexico because he can't hire enough programmers in this country. (The fact that he can hire experienced programmers in that country for one-fifth of what it costs in the U.S. is also a motivating factor, something he conceded only when we challenged him about it.) In another conversation, General Bill Thurman, commander of the Air Force's Aeronautical Systems Division, told us that recruiting, training, and keeping qualified programmers is becoming a problem of major proportions.

Compounding the problem is the simple fact that enrollments in university computer science programs continues to dip. For the first time in years, smaller colleges have openings for prospective computer science majors. Larger universities are still filling their slots, but they are seeing a dramatic decrease in interest in computer science. Last year, for instance, the University of California at Berkeley had 11 applicants for each computer science opening. Next year, one university spokesman told us a few days ago, the university expects each opening to have as few as five applicants.

Although the doctrine of supply-and-demand may provide short-term benefits for many programmers, the long-term consequences might be more detrimental. If nothing

else, the pressure necessitated by competitive schedules will have an adverse impact on the sanity of many programmers. That's the bad news. The good news is that programmers will be in a better bargaining position when it comes to defining their workplace because those companies that are able to attract top-flight programmers will be the companies who get the best software products out first.

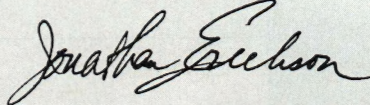
Working in real-time environments usually requires specialized development tools and sometimes specialized operating systems. The variety and complexity of real-time operating systems, however, may be foreign to those used to working with more mainstream, conventional operating systems. (The main distinction between real-time operating systems and more familiar one is in the way the real-time OS handles the task scheduling and priorities that enable predictable response time to events.)

We're of the opinion that real-time, embedded system development is an area with exciting potential. One reason for believing this is reflected in the growth in the embedded controller market itself.

By 1991, it is estimated that more than 800 million controllers will be shipped annually.

In practical terms, this has led to real technological innovation.

These emerging hardware platforms will give programmers the chance to create all kinds of new applications, but those programmers may first have to become familiar with new kinds of tools.



Jonathan Erickson
editor-in-chief

Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Editorial

Editor-in-Chief	Jonathan Erickson
Editor	Tyler Sperry
Managing Editor	Monica E. Berg
Associate Editor	Ron Copeland
Assistant Editor	Sara Noah Ruddy
Technical Editors	Allen Holub Richard Relp Kent Porter
Contributing Editors	Stan Krute Martin Tracy Rhoda Simmons Carol Dondrea Kevin Shafer
Copy Editors	Michael Swaine
Editor-at-Large	
Art/Production	
Director	Larry L. Clay
Art Director	Michael Hollister
Assoc. Art Director	Joe Sikoryak
Technical Illustrator	Barbara Mautz
Typesetter	Mary E. Lopez
Cover Photographer	Michael Carr
Circulation	
Circulation Director	Maureen Kaminski
Asst. Circulation Manager	Andrea Weingart
Newsstand Manager	Sarah Frisbie
Direct Marketing	Kathleen Shay
Fulfillment Coordinator	Francesca Martin
Administration	
Vice President of	
Finance and Operations	Kate Wheat
Business Manager	Betty Arsene
Accounts Payable Supv.	Mayda Lopez-Quintana
Accts. Receivable Supv.	Laura DiLazzaro
Marketing/Advertising	
Director	Ferris Ferdon
Advertising Coordinator	Patricia Albert
Account Managers	see page 145
Publisher	
	Peter Hutchinson

Dr. Dobb's Journal of Software Tools (USPS 307690) is published monthly, with two special issues per year by M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; 415-366-3600. Second-class postage paid at Redwood City and at additional entry points. DDJ is published under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a nonprofit corporation.

Article Submissions: Send manuscripts and disk (with article and listings) to the Associate Editor.

DDJ on CompuServe: Type GO DDJ

Address Correction Request: Postmaster: Send Form 3579 to Dr. Dobb's Journal, P.O. Box 3713, Escondido, CA 92025. **ISSN 0888-3076**

Customer Service: For subscription problems call: outside CA 800-321-3333; in CA 619-485-9623 or 566-6947. For book/software orders call 415-366-3600.

Subscriptions: \$29.97 per 1 year; \$56.97 for 2 years. Canada and Mexico add \$28 per year airmail or \$11 per year surface. All other countries add \$32 per year airmail. Foreign subscriptions must be prepaid in U.S. funds drawn on a U.S. bank. For foreign subscriptions, TELEX: 752-351.

Foreign Newsstand Distributor: Worldwide Media Service Inc., 386 Park Ave. South, New York, NY 10016; 212-686-1520 TELEX 620430 (WUI).

Entire contents copyright © 1988 by M&T Publishing, Inc., unless otherwise noted on specific articles. All rights reserved.



M&T Publishing Inc.

Chairman of the Board	Otmar Weber
Director	C. F. von Quad
President	Laird Foshay
Vice President of	
Publishing	William P. Howard

PolyAWK™ – The Toolbox Language™.

For C, Pascal, Assembly & BASIC Programmers.

We call PolyAWK our "toolbox" language because it is a general-purpose language that can replace a host of specialized tools or programs. You will still use your standard language (C, Pascal, Assembler or other modular language) to develop applications, but you will write your own specialized development tools and programs with this versatile, simple and powerful language. Like thousands of others, you will soon find PolyAWK to be an indispensable part of your toolbox.

A True Implementation Under MS-DOS

Bell Labs brought the world UNIX and C, and now professional programmers are discovering AWK. AWK was originally developed for UNIX by Alfred Aho, Richard Weinberger & Brian Kernighan of Bell Labs. Now PolyAWK gives MS-DOS programmers a true implementation of this valuable "new" programming tool. PolyAWK fully conforms to the AWK standard as defined by the original authors in their book, *The AWK Programming Language*.

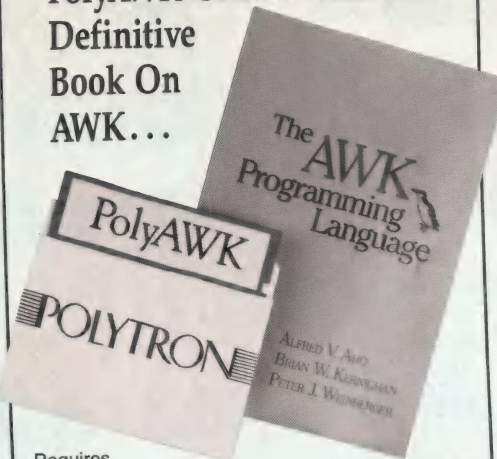
A Pattern Matching Language

PolyAWK is a powerful pattern matching language for writing short programs to handle common text manipulation and data conversion tasks, multiple input files, dynamic regular expressions, and user-defined functions. A PolyAWK program consists of a sequence of patterns and actions that tell what to look for in the input data and what to do when it's found. PolyAWK searches a set of files for lines matched by any of the patterns. When a matching line is found, the corresponding action is performed. A pattern can select lines by combinations of regular expressions and comparison operations on strings, numbers, fields, variables, and array elements. Actions may perform arbitrary processing on selected lines. The action language looks like C, but there are no declarations, and strings and numbers are built-in data types.

Saves You Time & Effort

The most compelling reason to use PolyAWK is that you can literally accomplish in a few lines of code what may take pages in C, Pascal or Assembler. Programmers spend a lot of time writing code to perform simple, mechanical data manipulation — changing the format of data, checking its validity, finding items with some property, adding up numbers and printing reports. It is time consuming to have to write a special-purpose program in a standard

PolyAWK Comes With The Definitive Book On AWK...



Requires MS-DOS 2.0 or above & 256K RAM.

\$99

When you order PolyAWK you receive a copy of *The AWK Programming Language* written by the authors of the original UNIX-based AWK. The book begins with a tutorial that shows how easy AWK is to use, followed by a comprehensive manual. Because PolyAWK is a complete implementation of AWK as defined by the book's authors, you will use this book as the manual for PolyAWK.

You can purchase PolyAWK and the book, *The AWK Programming Language*, for \$99. If you already have the book, you can order PolyAWK software only for \$85, which is \$14 off the regular \$99 purchase price. (The book serves as the User's Manual, so you should already have a copy of the book if you are ordering the software only.)

PolyShell Bonus!

PolyShell gives you 57 of the most useful UNIX commands and utilities under MS-DOS in less than 20K. You can still use MS-DOS commands at any time and exit or restart PolyShell without rebooting. MS-DOS programmers — discover what you have been missing! UNIX programmers — switch to MS-DOS painlessly! PolyShell and PolyAWK are each \$99 when ordered separately. Save \$50 by ordering the PolyShell + PolyAWK combination package for \$149. *Not copy-protected.*

30-Day Money Back Guarantee

Credit Card Orders:

1-800-547-4000

Ask for Dept. DDJ

Send Checks and P.O.s To:

POLYTRON Corporation

1700 NW 167th Place, Beaverton, OR 97006
(503) 645-1150 — FAX: (503) 645-4576

language like C or Pascal each time such a task comes up. With PolyAWK, you can handle such tasks with very short programs, often only one or two lines long.

Prototype With PolyAWK, Translate To Another Language

The brevity of expression and convenience of operations make PolyAWK valuable for prototyping even large-sized programs. You start with a few lines, then refine the program, experimenting with designs by trying alternatives until you get the desired result. Since programs are short, it's easy to get started and easy to start over when experience suggests a different direction. PolyAWK has even been used for software engineering courses because it's possible to experiment with designs much more readily than with larger languages. It's straightforward to translate a PolyAWK program into another language once the design is right.

Very Concise Code

Where program development time is more important than run time, AWK is hard to beat. These AWK characteristics let you write short and concise programs:

- The implicit input loop and the pattern-action paradigm simplify and often entirely eliminate control flow.
- Field splitting parses the most common forms of input, while numbers and strings and the coercions between them handle the most common data types.
- Associate arrays use ordinary strings as the index in the array and offer an easy way to implement a single-key database.
- Regular expressions are a uniform notation for describing patterns of test.
- Default initialization and the absence of declarations shorten programs.

Large Model Implementation

PolyAWK is a large model implementation and can use all of available memory to run big programs or read files greater than 64K.

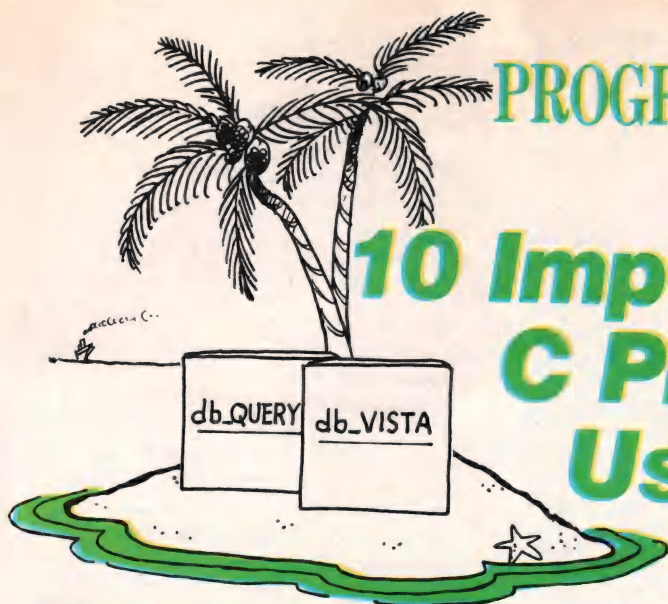
Math Support

PolyAWK also includes extensive support for math functions such as strings, integers, floating point numbers and transcendental functions (sin, log, etc.) for scientific applications. Conversion between these types is automatic and always optimized for speed without compromising accuracy.

POLYTRON

High Quality Software Since 1982

CIRCLE NO. 191 ON READER SERVICE CARD



PROGRAMMER'S PARADISE PRESENTS

10 Important Reasons C Programmers Use db_Vista from **RAIMA**TM CORPORATION

RAIMA & Programmer's Paradise offer you the best value on the tools you need.

1. It's written in C.

Clearly the growing language of choice for applications that are fast, portable and efficient.

2. It's fast — almost 3 times faster than a leading competitor.

Fast access that comes from the unique combination of the B-tree indexing method and the "network" or direct "set" relationships between records.

3. It's flexible.

Because of db_VISTA's combination of access methods, you can program to your application needs with ultimate design flexibility. Use db_VISTA as an ISAM file manager or to design database applications. You decide how to optimize run-time performance. No other tool gives you this flexibility without sacrificing performance.

4. It's portable.

db_VISTA operates on most popular computers and operating systems like UNIX, MS-DOS and VMS. You can write applications for micros, minis, or even mainframes.

5. Complete Source Code available.

The entire C Source Code is available so you can optimize performance or port to new environments yourself.

6. It uses space efficiently.

db_VISTA lets you precisely define relationships to minimize redundant data. It is non-RAM resident; only those functions necessary for operation become part of the run-time program.

7. Royalty free run-time.

Whether you're developing applications for yourself or for thousands, you pay for db_VISTA or db_QUERY only once. If you currently pay royalties to someone else for your hard work, isn't it time you switched to royalty-free db_VISTA?

8. SQL-based db_QUERY

Add Raima's new C-linkable, SQL-based, ad hoc query and report-writing companion product to provide a simple relational view of your db_VISTA applications.

9. Free tech support.

60 days of free technical and application development support for every Raima product.

10. Upward database compatibility

Start out with file management in a single-user PC environment — then move up to a multi-user LAN or a VAX database application with millions of records.

db_VISTATM Features

- ◆ Multi-user support allows flexibility to run on local area networks
- ◆ File structure is based on the B-tree indexing method
- ◆ Transaction processing assures multi-user consistency
- ◆ File locking support provides read and write locks
- ◆ SQL-based db_QUERY is linkable
- ◆ File transfer utilities included for ASCII, dBASE optional
- ◆ Royalty-free run-time distribution
- ◆ Source Code available
- ◆ Data Definition Language for specifying the content and organization of your files
- ◆ Interactive database access utility
- ◆ Database consistency check utility

File Management Record and File Sizes

- ◆ Maximum record length limited only by accessible RAM
- ◆ Maximum records per file is 16,777,215
- ◆ Maximum file size limited only by available disk storage
- ◆ Maximum of 256 index and data files
- ◆ Key length maximum 246 bytes
- ◆ No limit on number of key fields per record
- ◆ No limit on maximum number of fields per record

Operating System & Compiler Support

- ◆ Operating systems: MS-DOS, PC-DOS, UNIX, XENIX, UNOS, ULTRIX, Microport, VMS
- ◆ C compilers: Lattice, Microsoft, IBM, DeSmet, Aztec, Computer Innovations, Turbo C, XENIX and UNIX

db_VISTA or db_QUERY	List
Single user	\$195
Single user w/Source	\$495
Multi-user	\$495
Multi-user w/Source	\$990

Order Now

db_VISTA from Raima Corporation, put it to work in your application program.

Call Toll-Free Today!

1-800-445-7899
In NY: 914-332-4548

Programmer's
ParadiseTM

A Division of Hudson Technologies, Inc.
42 River Street, Tarrytown, NY 10591





Call or write
for the latest catalog

Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!

Welcome to Paradise. The microcomputer software source that caters to your programming needs.

Discover the Many Advantages of Paradise...

- Lowest price guaranteed
- Latest versions
- Huge inventory, immediate shipment
- Knowledgeable sales staff
- Special orders
- 30-day money-back guarantee*

Over 500 brand-name products in stock — if you don't see it, call!

We'll Match Any Nationally Advertised Price.

	LIST OURS	
386 SOFTWARE		
386-TO-THE-MAX	75	66
ADVANTAGE 386 C OR PASCAL	895	839
DESQVIEW	130	115
FOXBASE +/386	595	459
HIGH C-386	895	839
MICROPORT SYS V/386 (COMPLETE)	799	679
MS WINDOWS/386	195	130
NDP C OR FORTRAN-386	595	553
PHARLAP 386/ASM/LINK	495	422
SCO XENIX SYS V 386 (COMPLETE)	1595	1279
VM/386	245	182
X-AM	595	549
ARTIFICIAL INTELLIGENCE		
ARITY STANDARD PROLOG	95	80
MULISP-87 INTERPRETER	300	199
PC SCHEME	95	86
TURBO PROLOG	100	69
TURBO PROLOG TOOLBOX	100	69
ASSEMBLERS/LINKERS		
ADVANTAGE DISASSEMBLER	295	279
MS MACRO ASM (DOS OR OS/2)	150	99
OPTASM	195	172
PASM86	195	115
PLINK86PLUS	495	279
BASIC		
DB/LIB	139	121
FINALLY!	99	90
FLASH-UP	89	80
MACH 2	75	66
MS BASIC COMP. 6.0 (DOS OR OS/2)	295	189
MS QUICKBASIC	99	69
QUICKPAK	69	60
QUICKWINDOWS W/SOURCE	99	90
TRUE BASIC	100	80
TURBO BASIC	100	69
TURBO BASIC TOOLBOXES	100	69

DBASE TOOLS

APPLICATION PLUS	499	279
DBASE III PLUS	695	399
DBASE TOOLS FOR C OR PASCAL	90	69
DBFAST	69	60
DEBUG III	195	181
FOX TOOL BOX	295	287
FRIENDLY FINDER	99	90
GENIFER	395	282
HI-SCREEN XL	149	129
QUICK ENTRY	99	90
R&R	150	139
REPORT PLUS	120	131
THE DOCUMENTOR	295	249
TOM RETTIG'S LIBRARY	100	80
UI PROGRAMMER	295	249

C COMPILERS		
LATTICE C	500	272
MICROSOFT C (DOS OR OS/2)	450	285
QUICK C	99	69
TURBO C	100	69

C INTERPRETERS		
C-TERP	298	232
INSTANT C	495	384
RUN/C	120	85
RUN/C PROFESSIONAL	250	159

C LIBRARIES		
C ASYNCH MANAGER	175	137
C-FOOD SMORGASBORD	150	97
C TOOLS PLUS/5.0	129	101
C UTILITY LIBRARY	185	125
ESSENTIAL COMMUNICATIONS	185	125
COMMUNICATIONS PLUS	250	199
GREENLEAF C SAMPLER	95	69
GREENLEAF COMM LIBRARY	185	125
GREENLEAF FUNCTIONS	149	137
MULTI-C	395	215
PFORCE	198	169
RESIDENT C W/SOURCE	295	279
TIMESLICER	129	101
TURBO C TOOLS		
COBOL		
E-Z PAGE	295	269
MICRO FOCUS		
COBOL/2	900	733
COBOL/2 TLSET	900	733
PC-CICS	1500	1189
LEVEL II COBOL	349	282
PERSONAL COBOL	149	119
MICROSOFT COBOL	700	452
MICROSOFT SORT	195	130
OPT-TECH SORT	149	105
REALICS	995	799
REALIA COBOL	995	794
W/REALMENU	1145	899
RM/COBOL	950	763
RM/COBOL-85	1250	999
RM/SCREENS	395	339
SCREENIO	400	382
COMMUNICATIONS		
ASCOM IV	195	177
CARBON COPY PLUS	195	142
CO-SESSION (2 USER)	249	227
SUPPORT	175	157
APPLICATION	125	116
PTCL	50	45
SIDETALK	120	90

DEBUGGERS		
ADVANCED TRACE-86	175	121
PERISCOPE I	345	282
PERISCOPE II	175	141
PERISCOPE III 8 MHZ	1095	899
PERISCOPE III 10 MHZ	1195	979
PFIX 86 PLUS	395	215

DISK/DOS/KEYBOARD UTILITIES		
ADVANCED NORTON UTILITIES	150	101
COMMAND PLUS V. 2.0	80	70
DISK OPTIMIZER	75	56
FETCH	150	55
NORTON COMMANDER	75	56
PC TOOLS DELUXE	80	70
PDISK	145	107
VFEATURE	495	

EDITORS		
BRIEF	195	CALL
W/DBRIEF	275	CALL
EMACS	295	268
EPSILON	195	151
KEDIT	150	120
MKS VI	75	66
MULTI-EDIT	99	90
NORTON EDITOR	75	70
PC/EDIT+	295	269

FILE MANAGEMENT		
BTRIEVE	245	185
XTREVIEW	245	188
REPORT OPTION	145	109
BTRIEVE/N	595	455
XTREVIEW/N	595	459
REPORT OPTION/N	345	279
CBTREE	159	141
C-TREE	395	318
R-TREE	295	241
C-TREE/R-TREE BUNDLE	650	523
D-TREE	395	CALL
DBC III	250	172
DBC III PLUS	750	599
DB VISTA OR DB QUERY	495	CALL
SINGLE USER W/SOURCE	495	CALL
MULTIUSER	495	CALL
MULTIUSER W/SOURCE	990	CALL
INFORMIX PRODUCTS	CALL	CALL
XQL	795	599

FORTRAN COMPILERS		
LAHEY FORTRAN F77-EM/16	695	629
LAHEY PERSONAL FORTRAN 77	95	86
MS FORTRAN (DOS OR OS/2)	450	285
RM/FORTRAN	595	479

FORTRAN LIBRARIES/UTILITIES		
DIAGRAM'ER OR DOCUMENT'ER	129	115
GRAFMATIC OR PLOTMATIC	135	119
MAGUS NUMERICAL ANALYST	295	252
MATHPAC	495	445
SPINDRIFT LIBRARY	149	135
SSP/PC	350	272

GRAPHICS		
ADVANTAGE GRAPHICS (C)	250	229
ESSENTIAL GRAPHICS	299	229
W/SOURCE	598	509
GSS GRAPHIC DEV. TOOLKIT	495	399
HALO '88	325	229
HALO '88 (5 MICROSOFT LANG.)	595	399
METAWINDOW PLUS	275	232
TURBOWINDOW/C	95	80
TURBO HALO (FOR TURBO C)	99	80

MODULA-2		
LOGITECH MODULA-2	99	81
COMPILER PACK	249	199
DEVELOPMENT SYSTEM	169	141
TOOLKIT	75	59
SOLID B+ TOOLBOX	99	89
STONYBROOK MODULA-2	195	179
W/UTILITIES	345	299

OBJECT-ORIENTED PROGRAMMING		
ACTOR	495	423
ADVANTAGE C++	495	479
PFORCE++	395	215
SMALLTALK/V	100	85
APPLICATION PACKS	50	45
SMALLTALK/V286	200	169

OPERATING SYSTEMS		
MICROPORT DOS MERGE	149	129
MICROPORT SYS V/AT	549	599
SCO XENIX SYSTEM V (COMP.)	1295	999
WENDIN-DOS	99	80
OTHER MICROPORT.SCO.		
WENDIN PRODUCTS		

CALL CALL

LIST OURS

PASCAL COMPILERS		
MICROSOFT PASCAL (DOS OR OS/2)	300	189
PASCAL-2	229	CALL
TURBO PASCAL	100	69
TURBO PASCAL DEV. LIB.	395	289
BORLAND ADD-ONS		CALL CALL

FEATURED PRODUCTS

NORTON EDITOR — Fast editor with all the essential features at the right price. Split-screen edit, auto-indent for Pascal and C, word wrap, word action, mouse support, easily customized.

HALO '88 — Newly enhanced version of this graphics subroutine library now supports over 140 hardware devices. Designed to support the PS/2 series and VGA. Compatible with 18 programming languages. 20 fonts are now included.

SCO VP/IX — Extension of the XENIX operating system, enables users to run DOS software "off the shelf" as a task under SCO XENIX on Intel 80386-based computers.

GREENLEAF DATAWINDOWS FOR OS/2 — This version takes full advantage of OS/2 virtual memory and multitasking. Completely rewritten assembler code for OS/2, not just a re-compile. Works in a multi-thread situation, with windows that can be made swappable when you wish. Create logical windows, easy menus, list boxes, transaction data entry. Source code included. Supports Microsoft and Lattice C.

TURBO PASCAL ADD-ONS

ASCII TURBO GHOST WRITER	99	80
STARTER COMPLETE	289	262
DOS/BIOS & MOUSE TOOLS	75	70
FLASH-UP	89	80
METRABYTE DATA ACQ. TOOLS	100	90
SCREEN SCULPTOR	125	96
SYSTEM BUILDER	150	131
IMPEX	100	90
REPORT BUILDER	130	116
T-DEBUG PLUS V. 4.0	45	39
W/SOURCE	90	80
TURBO ASM	99	70
TURBO ASYNCH PLUS	129	101
TURBO GEOMETRY LIBRARY	100	90
TURBO HALO	99	80
TURBO MAGIC	99	90
TURBO POWER TOOLS PLUS	129	101
TURBO POWER UTILITIES	95	79
TURBO PROFESSIONAL 4.0	99	80
TURBO WINDOW/PASCAL	95	80

LIST: \$395 Ours: \$279

LIST: Two-users: \$495 Ours: \$399

LIST: Unlimited users: \$995 Ours: \$799

TURBO PASCAL ADD-ONS

ASCII TURBO GHOST WRITER

STARTER COMPLETE

DOS/BIOS & MOUSE TOOLS

FLASH-UP

METRABYTE DATA ACQ. TOOLS

SCREEN SCULPTOR

SYSTEM BUILDER

IMPEX

REPORT BUILDER

T-DEBUG PLUS V. 4.0

W/SOURCE

TURBO ASM

TURBO ASYNCH PLUS

TURBO GEOMETRY LIBRARY

TURBO HALO

TURBO MAGIC

TURBO POWER TOOLS PLUS

TURBO POWER UTILITIES

TURBO PROFESSIONAL 4.0

TURBO WINDOW/PASCAL

SCREENS/WINDOWS

C-SCAPE

CURSES W/SOURCE

GREENLEAF DATA WINDOWS

HI-SCREEN XL

JYACC FORMAKER

JYACC IAM

MICROSOFT WINDOWS

MS WINDOWS DEVELOPMENT KIT

PANEL PLUS

PANEL/PC OR /TC

SCREENSTAR W/SOURCE

VIEW MANAGER

VITAMIN C

VC SCREEN

WINDOWS FOR DATA

W/SOURCE

FREE T-SHIRT

When you buy two or more products before Aug. 1, 1988.

Net accounts excluded.

One per customer.

While supply lasts.

ADDITIONAL PRODUCTS

ADVANTAGE VCMs

BABY/36 (RPG II)

BASTOC

DAN BRICKLIN'S DEMO PROGRAM

DEMO PROGRAM II

DB2C

FLOW CHARTING II

MAGIC PC

MKS PCS

MKS-SQPS

MKS TOOLKIT

MS OS/2 PROG. TOOLKIT

NORTON GUIDES

PC-LINK

POLYMAKE

POLYTRON PVCS

PRE-C

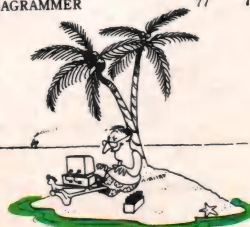
SOURCE PRINT

TREE DIAGRAMMER

Terms and Policies
 • We honor MC, VISA, AMERICAN EXPRESS
 No surcharge on credit card or C.O.D. Prepayment by check. New York State residents add applicable sales tax. Shipping and handling \$3.95 per item within the U.S., sent UPS ground. Rush and international service available. Call for prevailing rates.
 • Programmer's Paradise will match any current nationally advertised price with equivalent terms for the products listed in this ad.
 • Prices and Policies subject to change without notice.
 • Hours 9AM EST - 7PM EST
 • Mail Orders include your phone number
 *Ask for details. Some manufacturers will not allow returns once disk seals are broken.
 Dealers and Corporate Buyers — Call for special discounts and benefits!

1-800-445-7899
In NY: 914-332-4548
 Customer Service:
914-332-0869
 International Orders:
914-332-4548
 Telex: 510-601-7602

Programmer's Paradise
 A Division of Hudson Technologies, Inc.
 42 River Street, Tarrytown, NY 10591
 CIRCLE NO. 195 ON READER SERVICE CARD



LETTERS



Pascal, C, and Row Storage of Arrays

Dear DDJ,

In his "Structured Programming" column in the March 1988 issue, Kent Porter points out that two-dimensional arrays can be implemented as an array of pointers to rows instead of the usual contiguous array of rows (where a row is itself an array of data elements). Besides overcoming the 8088 64K segment restriction, this programming technique is commonly used for arrays where the rows have different lengths or are of uncertain length and number.

The best known example for C programmers is the processing of the command line. The hidden startup code splits the command line into words and allocates space for and fills in an array of pointers to the words. It then passes the address of the pointer array to the function *main()*. Other examples are given in Kernighan and Ritchie's *The C Programming Language*.

Porter's implementation of this technique in Pascal, while a service for Pascal programmers, also illustrates some contrasts between Pascal and C. In Pascal, array element references change their syntax, after auxiliary type declarations, from *a(I,J)* to the more awkward *A(I).col(J)*. To convert a program to use large matrices with the alternate storage method, all array references would have to be modified. In C, only the allocation function would have to be changed. Arrays references would keep the form *A[I][J]* in spite of the change in stor-

age method since the compiler handles such internal details. This is illustrated by the following short program which prints the command line as a two-dimensional array of characters.

```
main(int n, char **v) {
    register int i, j;
    for (i = 0; v[i]; i++) {
        for (j = 0; v[i][j]; j++)
            putchar(v[i][j]);
        putchar('\n');
    }
}
```

Admittedly, however, if the C programmer had used explicit pointer references for the sake of 'efficiency,' then the conversion effort might be even greater than with Pascal.

Terry J. Reedy
Los Angeles, CA

Fortran Implementation of Hugemats

Dear DDJ,

I realize that Fortran is no longer fashionable as a computer language,

```
c Fortran implementation of 'hugemats'

c
program hugemat
implicit integer*2 (a - z)

parameter (mrow=250, mcol=300)

dimension a(mrow,mcol), b(mrow,mcol),
           c(mrow,mcol)

call acquir (a, mrow, mcol)
call acquir (b, mrow, mcol)

do 20 i = 1, mrow
  do 10 j = 1, mcol
    c(i,j) = a(i,j) + b(i,j)
  10 continue
20 continue
print 30, 'Proof:'
print 40, '1[1,1]', a(1,1)
print 40, 'b[1,1]', b(1,1)
print 40, 'c[1,1]', c(1,1)
print 30, ' '
print 40, 'a[mrow,mcol]', a(mrow,mcol)
print 40, 'b[mrow,mcol]', b(mrow,mcol)
print 40, 'c[mrow,mcol]', c(mrow,mcol)
30 format (' ',a)
40 format (' ',a,' = ',i5)
end

subroutine acquir (array, n1, n2)
implicit integer*2 (a - z)
dimension array(n1,n2)

do 20 i = 1, n1
  do 10 j = 1, n2
    array(i,j) = i * 10 + j
  10 continue
20 continue
return
end
```

Example 1: A Fortran Implementation of "hugemats"

but—without casting aspersions on Borland's dialect of Pascal—I would like to submit the example below as a substitute for Kent Porter's Turbo Pascal program *hugemats* (DDJ, March 1988). Apart from the compiler directives, it is a portable Fortran 77 program and, although without comments, its intent is more immediately obvious.

C.B. Chapman
London, Ontario

And Debate Goes On

Dear DDJ,

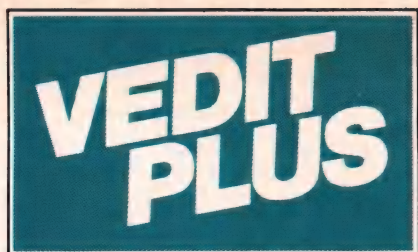
I have followed the "Turbo C vs. Quick C" debate with some interest. Availability? Quick C has been on sale at my local branch of B. Dalton for weeks. Turbo C was rushed to market earlier (but still late) and full of bugs.

Few people have focused upon the single most important difference between the two compilers—debugging. In my many years as a professional programmer I have learned that finding runtime bugs is by far the most difficult and time consuming part of programming. Finding syntax errors at compile time is normal, but not significant. Why has Borland never produced a source-level debugger like CodeView? There are plenty out there that Borland could simply buy and customize. I think that this is the single greatest weakness of all Borland products.

Colin J. Davies
Granada Hills, CA

Dear DDJ,

I was reading with amusement your recent exchanges about the relative merits of the Quick C and Turbo C compilers. Since I have bought and used both, and since I returned my Quick C compiler for a refund, I have some opinions about them. My version of Quick C was 1.0, and my Turbo C is 1.5. I have a 4,800 line program that had already trashed my Let's C source-level debugger and my DOS-Debug utility, and it did the same for Quick C's debugger. The Quick C debugger developed a case of frozen screen with funny-looking symbols and music notes appearing at random on what



#1 PROGRAMMABLE EDITOR

Call 1-800-45-VEDIT for
FREE Fully Functional Demo Disk

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today. You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. \$185.

Special: VEDIT (single file, no windows) for CP/M—\$49.

- Fully Network Compatible
- Call for XENIX-286 version
- 30 Day Money-back guarantee

Compare Features and Speed

	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	Unlimited
Multiple file editing	20 +	2	No	20 +
Windows	20 +	2	No	20 +
Macro execution window	No	No	No	Yes
Pop-up menus	No	No	No	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Automatic processing of				
Compiler errors	Yes	No	No	Yes
"Cut and paste" buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
Convert to/from WordStar	No	No	No	Yes
On-line calculator	No	No	No	Yes
Configurable Keyboard	Hard	No	Hard	Easy
43 line EGA support	Yes	No	No	Yes
Manual size/index	250/No	42/no	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min	34 sec	1:07 min	6 sec
Pattern matching search	20 sec	Cannot	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec



VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc. MultiLink and PC-MOS/386 are trademarks of The Software Link, Inc. CP/M and FlexOS are trademarks of Digital Research. MS-DOS is a trademark of Microsoft.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others.
*Demo disk is fully functional, but does not readily write large files.

CIRCLE NO. 109 ON READER SERVICE CARD

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299, TELEX 701821

was supposed to be the display of my program executing.

About the Quick C documentation: Yes, indeed, it does have excellent C documentation, although the Turbo C documentation really is not that bad compared with, say, *The C Programming Language* of K and R. But the documentation for the Quick C debugger is almost nonexistent and what is there is cryptic and unusable. Let someone cleverer than me find out how to single-step the debugger from the documentation; or, how to stop the debugger once it is started in "display steps in execution" mode. There are no answers anywhere in the manuals, nor even references in the index on the debugger.

Since the main reason that I bought Quick C was to use the debugger, I have some reasons to dislike what I got: first of all, the debugger is crippled. It only accepts programs that have just been compiled, and excludes re-loading and debugging previously compiled programs. You can only compile in medium

model if you want the debugger to work, even if you have a program that only works in huge model (I hope C beginners realize that large programs have to use special "memory models" to work properly on the PC), and Quick C does not support huge model compilation. And, let me tell you, you can write a will and fill out your income tax while Quick C is compiling and then loading the debugger. With my 4,800-line program, the debugger took over a half hour to compile and load. If you have ever used the Let's C compiler and source debugger, you know that Quick C makes Let's C look good on that basis.

Do you know what the underground bargain C compiler of this year is? It's the Mix Power C compiler. For under \$25 with shipping, it is one heck of a good compiler, and the manual is worth the price of admission. It's C preprocessor will substitute macro arguments into the middle of strings for you (so you can macroize your *printf* ("debug message with dubbed parameter") state-

ments using your *h* file); and, while it has a problem with newlines inserted in the middle of parameters supplied to the macro-processor, it is otherwise quite stable and handles really large programs (in large model, not huge). While Turbo C is faster and can handle huge model programs, its macro preprocessor will not substitute arguments into the middle of strings for you, instead the strings resulting from the macro-expansion will contain the macro parameter name.

To be fair, the Turbo C Windows package is worth the cost of the compiler, and the Power C business math and graphics package is definitely a worthwhile acquisition. Turbo C has to be the world's fastest screen writer, both in windows mode and standard *printf* to *stdout* and *stderr* mode. Both Turbo C and Power C have optional run-time stack overflow tests automatically inserted into compiled code.

Since C programs work in mysterious ways and can and do overwrite memory and go into recursive bugs, the stack test is a minimal safety net that is really needed. I only wish that the enumeration type of C could be used to force range tests Pascal-style in compiled code. That way, while debugging I could *enum* some of my important variables and specify their lower and upper bounds in the *enum* set, knowing that the compiler would insert correctness tests wherever I did C arithmetic with these variables. As matters presently stand, the compilers refuse to let me do arithmetic with these variables, and don't seem to enforce bounds on their values by run-time checking.

Victor Schneider
Brighton, MA 02146

DDJ



Unbeknownst to his colleagues, Gil had reached the end of his interrupt chain.

PVCS

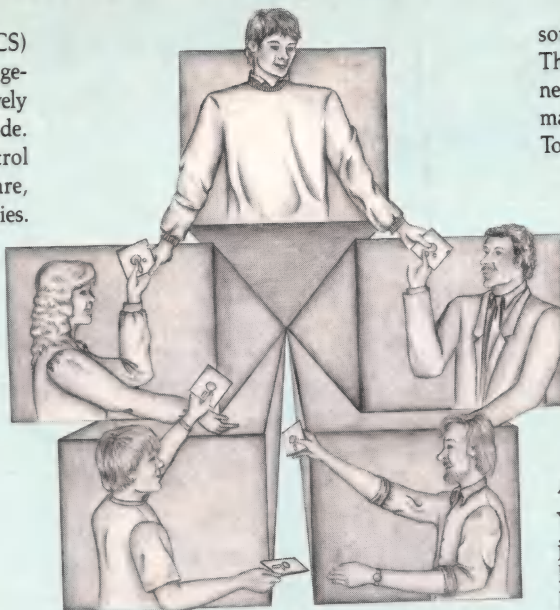


The Number One Source Code Control System.

The POLYTRON Version Control System (PVCS) simplifies and automates Configuration Management so programmers and managers can effectively control the revisions and versions of source code. PVCS is the most widely used change control product and is used by the leading software, aerospace, manufacturing and service companies.

"In terms of features, PVCS provides everything necessary to a large multi-programmer project — more than any other package reviewed. No restrictions are placed in the development environment and all aspects of operation can be customized for specific project needs."

*PC Tech Journal
September 1987*



source files, libraries, object code and other files. The levels of security can be tailored to meet the needs of nearly every project. PVCS works on all major LANs including 3Com, Novell and the IBM Token Ring Network.

"PVCS has helped us maintain nearly 90 programs and utilities. Without it we would not have the quality of our upcoming release of NetWare."

*Jonathan Richey
Manager, NetWare Utilities
Novell*

Unmatched Flexibility

- Storage & Retrieval of Multiple Revisions of Source Code
- Maintenance of a Complete History of Changes
- Control of Separate Lines of Development (Branching)
- Resolution of Access Conflicts
- Optional Merging of Simultaneous Changes
- Release and Configuration Control
- Project Activity Reports
- Management Reports
- Command or Menu Interface

Project Control

PVCS maintains individual archives of all project components in your system — source code modules, data files, documentation and even object code libraries. These "source documents" can be written in any language or multiple languages.

Fast Retrieval of Revisions

PVCS uses "reverse delta storage" which saves disk space and speeds retrieval of versions of any file in the project database. A delta is the set of differences between any revision and the previous revision. PVCS can rapidly recreate complete versions of any file whether it is the most recent revision of a module or the original version of the entire project. Differences are automatically detected and stored.

A Practical Necessity for LANs

While important for single-programmer projects, PVCS is absolutely essential for multiple-programmer projects and LAN-based development efforts. In a LAN environment, source code files are simply too easy to change. Because any change to any file can have major ramifications, coordinating and keeping a record of changes is critical. Project leaders can determine, on a module-by-module basis, which programmers can access or modify

Once you standardize on PVCS, the archives used to track and monitor changes are interchangeable between any PVCS product.

Personal PVCS — Offers most of the power and flexibility of Corporate PVCS, but excludes the features necessary for multiple-programmer projects.

Corporate PVCS — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes multi-level branching to effectively maintain code when programs evolve on multiple paths.

Network PVCS — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project.

PVCS for VAX systems — Requires VMS. Uses the same interface and archive format as MS-DOS version. Supports branching and offers file locking and other security features for multiple-programmer projects.

Adopt PVCS on Your Existing Projects

You can obtain the benefits for your current project without disrupting development, regardless of how long your project has been under way. You can build PVCS archives from revisions stored in your present files or simply adopt PVCS from the current date.

PolyMake Reads PVCS Logfile Format

PolyMake, the leading Make utility, understands the structure of PVCS logfiles and is able to correctly determine the date and time of any revision. This prevents unnecessary operations that occur when the date and time of the complete project archive itself is used as with other make utilities.

	MS-DOS*	VMS		
	PC/XT/AT	Micro VAX II	VAX 7xx	VAX 8xxx
Personal PVCS	\$149			
Corporate PVCS	\$395			
Network PVCS	\$995**	\$4,950	\$9,500	\$10,500+
PolyMake	\$149			
Network PolyMake	\$447**	\$1,250	\$2,375	\$2,500+

**5 Station LAN License. Call for pricing on larger Networks.

TO ORDER:
1-800-547-4000
Dept. DDJ

Oregon & Outside USA call (503) 645-1150.
Send Checks, P.O.s to: POLYTRON
Corporation, 1700 NW 167th Place,
Beaverton, OR 97006

POLYTRON

High Quality Software Since 1982

CIRCLE NO. 190 ON READER SERVICE CARD



One Language For W

BBX[®] Specifications:

Ease-of-use—BBX[®] is the fastest, most powerful development tool available for business oriented program creation. Programmers can write code in minutes.

Execution time—BBX's partially compiled format provides enhanced execution speed.

Easy Maintenance—BBX[®] is an interactive programming language, with a trace facility and a full screen editor which makes program maintenance a snap.

Portability—BBX[®] runs under UNIX and other operating systems without recompilation.

Compatibility—BBX[®] is an enhancement of the Business BASIC language, an industry standard, giving you access to thousands of applications.

Supportability—Program maintenance utilities and complete documentation save considerable time and money. It lets you build and support applications easily.

Utilities—A complete set of BBX[®] utilities are provided for program and file management.

Conversion—A complete set of conversion utilities are provided with every BBX[®] package.

Features

Math Functions

- 14 place precision and computational accuracy
- Floating point conversion
- Task specified rounding precision
- Binary to decimal conversion
- Long function names
- Dynamic arrays
- String Functions**
 - Numeric to string conversion
 - String manipulation
 - No string length restriction
- I/O Functions**
 - Windowing
 - I/O mnemonics
 - Device independent verbs
 - X,Y cursor addressing
 - Masking
 - Soft key loads
 - No record length restrictions
 - BBX[®] file sizes are limited only to the size of the available media

File Structures

- INDEX
- KEYED

• MKEYED

- SERIAL
- SORT
- PROGRAM
- STRING

System Structure

- Multi-tasking - which provides record and file level locking
- Program overlay
- Public programming which provides:
 - Local variables
 - Dynamically called sub-programs
 - Argument passing
 - Automatic public program drop from memory at exit
 - Public program in memory lock option

Language Structure

- Interactive program development
- Online syntax checking
- Compound statements
- User defined functions
- Unlimited nesting
- Remote I/O lists
- Program self modification
- Case insensitive console mode
- Various debugging tools

BBX[®] Utility Set

- File Browse
- Create Data Bundle
- Calculator
- Clear Workspace
- Program Compare
- Copy File
- Define/Redefine File
- Directory Listing
- Erase File
- Generate Filelist
- Program List/Cross Reference
- Move File
- Program Renumbered
- Rename File
- File Resizer
- Execute O/S Shell Command
- Search and Replace Program
- Color & FUNC Key Setup
- Time/Date Examine/Set
- Utility Menu
- Visual Utility Interface
- BXSND/BXRVC conversion utilities

Its portability crosses all operating environments, and now its performance is crossing all oceans.

Around the world, the industry's best and brightest programmers are discovering the astonishing power that BBX[®] brings to Business BASIC. Write your program once, and have complete movement to MS/PC-DOS, OS/2, UNIX/XENIX, AIX, IX370 and VMS.

This year, over 50,000 copies of BBX[®] are performing throughout the United States, Canada, Europe, Asia and South America.

Commitment to innovation, development within industry standards and technological leadership have grown BBX[®] around the globe.

In 1988, aggressive marketing and uncompromising customer support will continue to compliment our success, and expand the BBX[®] standard among many of the world's most respected companies.

Get in touch with one of our world distributors, and feel the pulse of the power of BBX[®]!

Die Portabilität schlägt sämtliche, bisher bekannte und unbekannte, EDV-Umgebungen. Die Leistung überzeugt inzwischen die gesamte EDV-Industrie.

Weltweit entdecken die besten Software-Entwickler die erstaunliche Leistung von BBX[®], mit der Business BASIC bereichert wird. Die Anwendungen werden nur einmal entwickelt und laufen ohne Änderungen oder Anpassungen auf MS/PC-DOS, OS/2, UNIX/XENIX, AIX, IX370 oder VMS.

Mehr als 50.000 BBX[®]-Lizenzen stellen die Leistung in den USA, Canada, Europa, Asien und Süd-Amerika unter Beweis.

Zur Innovation nach Industrie-Standard Spezifikationen verpflichtet, und mit dem Ziel nach technologischer Führung, wächst BBX[®] um die Welt.

Mit aggressivem Marketing ohne Kompromisse im Bereich Kundenservice, wird der Erfolg von BBX[®] in 1988 fortgesetzt. Es steht auf sämtlichen Systemen namhafter Computerhersteller zur Verfügung und stellt seine Akzeptanz bei den anspruchsvollsten Anwendern unter Beweis.

Kontaktieren Sie unsere Vertretungen in aller Welt. Entdecken Sie die Schlagkraft von BBX[®]!

BBX[®] PROGRESSION/2[®] is available for Intel Based Computers, Altos, Arete, AT&T, PCS Cadmus, Computer Consoles, Convergent Technologies, Counterpoint/MultiTech, Cubix, Data General, Digital Equipment, Fortune, Honeywell, Hewlett Packard, ICL, Motorola, Nixdorf, Prime, Pyramid, Rexon, Sanyo, Sequent, Siemens, Texas Instruments, Unisys, and the IBM family of products. BASIS is continually adding new systems.



World Class Business.

Portable, il franchit tous les cadres d'opération, et sa performance traverse, maintenant, tous les océans.

Dans le monde, les meilleurs et les plus brillants programmeurs de l'industrie découvrent l'étonnante puissance que BB^x amène au BASIC des affaires. Ecrivez votre programme une seule fois, et accédez totalement à MS/PC-DOS, OS/2, UNIX/XENIX, AIX, IX370 et à VMS.

Cette année, plus de 50 000 copies de BB^x fonctionnent aux États-Unis, au Canada, en Europe, en Asie et en Amérique du Sud.

Un esprit constant d'innovation, un développement conforme aux normes de l'industrie, et une position de leader dans le domaine technologique, tels sont les atouts qui ont contribué à la croissance de BB^x dans le monde entier.

En 1988, un marketing dynamique et un appui inconditionnel à notre clientèle continueront à couronner notre réussite, et à étendre le standard BB^x à de nombreuses sociétés parmi les plus respectées au monde.

Contactez l'un de nos distributeurs mondiaux et découvrez la puissance de BB^x!

Su portabilidad traspasa todos los medios de operación y ahora su funcionamiento esta cruzando todos los océanos.

Los mejores y más brillantes programadores del mundo, están descubriendo la asombrosa potencia que BB^x ofrece al negocio BASIC. Escriba su programa una vez y tenga movimiento completo a MS/PC-DOS, OS/2, UNIX/XENIX, AIX, IX370 y VMS.

Este año, más de 50,000 copias de BB^x estan funcionando en Estados Unidos, Canadá, Europa, Asia y América del Sur.

Empeño de inovación, desarrollo en los estandards de la industria y superioridad tecnológica han hecho crecer a BB^x en todo el mundo.

En 1988, mercadotecnia agresiva y apoyo constante a nuestros clientes seguirán complementando el éxito y desarrollo de BB^x entre las compañías más respetadas del mundo.

Comuniquese con uno de nuestros distribuidores mundiales y sienta la potencia de BB^x!

World Distributors:

Edisa Hans Kirchhoff & Co. KG
Pingsbornstr. 25, 6200 Wiesbaden
TEL: (06122) 2016
FAX: (06122) 16505
TLX: 418-2563 edia d

Multieya
Torgeir Vraas Plass 5A
3044 Drammen Norway
TEL: (03) 83.86.05
FAX: (03) 89.02.53

J. P. Brown and Associates
780 Gordon Baker Road
Willowdale, Ontario M2H 3B4
TEL: (416) 494-0472

PI Informatique
8, rue Benjamin Constant
75019 Paris, France
TEL: (01) 40.05.10.85
TLX: 214.583
FAX: (01) 40.05.99.63

Softach Pty. Limited
10 Eileen Road, Blairgowrie
Randburg 2194, South Africa
TEL: (011) 787-8839
FAX: (011) 886-3890
TLX: 422669 S.A.

Risegold Pty. Ltd.
678 Paramatta Road
Croydon, N.S.W. 2132
Australia
TEL: (02) 799-6622
FAX: (02) 799-9090

Tempo Computadoras
Av. Americas #670
Guadalajara 44680
Jalisco, Mexico
TEL: 30-28-45
30-28-46
30-28-86

In the United States:
BASIS Incorporated
P.O. Box 20400
Albuquerque, New Mexico 87154
TEL: (505) 821-4407
FAX: (505) 821-1625

West Germany, The Netherlands
Austria, Switzerland
Denmark, Luxembourg, Belgium
England, Italy

Norway, Sweden, Finland,
Greenland, Iceland

Canada

France, Spain, Portugal

South Africa

Risegold Pty. Ltd.
86 Havelock Street
West Perth
Western Australia 6005
TEL: (09) 481-0607
FAX: (09) 481-3162

Infotai, S.A. de C.V.
Laguna de Mayran
No. 258 3 piso
Col. Anahuac C.P. 11320
Mexico, D.F.
TELS: 05 45 6730 al
05 45 6734

Australia

Mexico



BB^x PROGRESSION/2®, BB^x and BASIS Incorporated are trademarks and/or service marks of BASIS Incorporated, Albuquerque, New Mexico. All references to computer systems and software products contained within this advertisement recognize the trade and/or service marks of the corresponding manufacturer and holder of the trade and/or service mark.

CIRCLE NO. 90 ON READER SERVICE CARD

10% DISCOUNT
ANY ORDER BY A NEW CUSTOMER, ACCOMPANIED
BY THIS ADVERTISEMENT, QUALIFIES
FOR A 10% DISCOUNT!
Call TOLL FREE directly to our
Order Department.
1-800-423-1300
DDM

WRITING REAL-TIME PROGRAMS UNDER UNIX

by Bill Cramer

For most programmers, the expression "real-time with Unix" falls into the same category as "user-friendly yet powerful" and "self-documenting code." But just as there actually exist a few powerful, user-friendly products (which were no doubt written with self-documenting code), Unix is a realistic and practical operating system choice for many real-time or near-real-time applications.


Most applications fall into one of three broad classes—interactive (such as a word processor), batch mode (such as a payroll program), and real time (such as the module in a car that tells you the door is open). These applications perform best when running under an appropriate operating system and perform worst when running under an inappropriate OS—for instance, a payroll program that requires extensive user interaction for each of 19,000 employees would drive an accounting staff over the edge. Likewise, using a batch-mode operating system to read the temperature module on your car's engine might not tell you that the car has overheated until a few hours after the fact.

A few applications, however, need a subtle mix of different operating system characteristics. In a computer-automated assembly line, for example, each assembly workstation may have its own computer or embedded processor running under a real-time operating system. These real-time computers keep busy reading gauges, moving robot arms, turning switches on and off, and so on. A central computer that doesn't need to precisely control valves opening or the amount of torque a robot arm applies to the handle may preside over the individual workstations. That central computer has more global concerns—for instance, it may have to shut down the assembly line if an individual station doesn't have the parts required to build the product.

The central computer may intermittently receive input (how many parts it has processed and so on) from individual workstations. It may then send commands to other workstations to speed up or slow down the assembly line or to reroute a certain part to a different station. It may also sound an audible alarm to alert the foreman that a particular machine needs servicing.

The real-time OS in the workstation controller generally performs the work it was designed to

Bill Cramer is a software engineer with Tekn Kron Infoswitch, 1784 Firman Dr., Richardson, TX 75081, which builds Unix-based adjunct processors for PBX and central-office telephone switches. Most recently, he has been working in the area of software productivity.



```

if (!clock_opened)
{
    fd=open("/dev/clock", O_RDONLY | O_NDELAY);
    ioctl (fd, TCGETA, &clock_termio);
    clock_termio.c_cflag |= CLOCAL;
    clock_termio.c_lflag &= ICANON;
    ioctl (fd, TCSETA, &clock_termio);
    flags = fcntl (fd, F_GETFL, 0);
    fcntl (fd, F_SETFL, flags);
    clock_opened = TRUE;
}

/* set the VTIME delay to the
   clock_termio.c_cc[VMIN]

```

The best solution lies in understanding the nature of the operating system—its strengths and weaknesses—and working with the system instead of against it.

do very well; however, it was never really intended to write reports, maintain a database, or display the status of all the individual workstations on the factory floor. To perform these tasks, the central computer needs a general-purpose, widely used, widely supported OS such as Unix.

To understand how Unix serves both the real-time world and the general-purpose world, it is necessary to understand how real-time systems differ from time-sharing systems. Two common multitasking system features—process scheduling and process blocking—can illustrate how Unix differs from a real-time system.

The Scheduler

Multitasking operating systems include a *scheduler*, which has the responsibility of allocating CPU time among the various loaded processes. Even though both Unix and most real-time operating systems have schedulers, their basic algorithms differ sharply. A real-time system is *event-driven*, whereby the OS allocates CPU

time to processes that need to service events. (Events include new input data, a clock time-out, and so on.) The scheduler in Unix, on the other hand, is a time slicer; it attempts to give all processes a chance to run.

The system (or the user of the system) may assign a priority to each process in the system. For a real-time system, a high-priority process with outstanding events will have near-absolute control of the CPU; only a hardware interrupt can intrude on the process. A low-priority process will run only when the high-priority process can no longer run (for example, when it must wait on some new external event).

Under Unix, the OS may give a high-priority process a larger slice of CPU time than it does a low-priority process. However, when the high-priority process has used up its time slice, the OS suspends it and begins execution of another process.

Asynchronous and Synchronous System Requests

At any given time, a process is in one of three broad states: *running*, in which the process has control of the CPU; *runable*, in which the process would like to run but

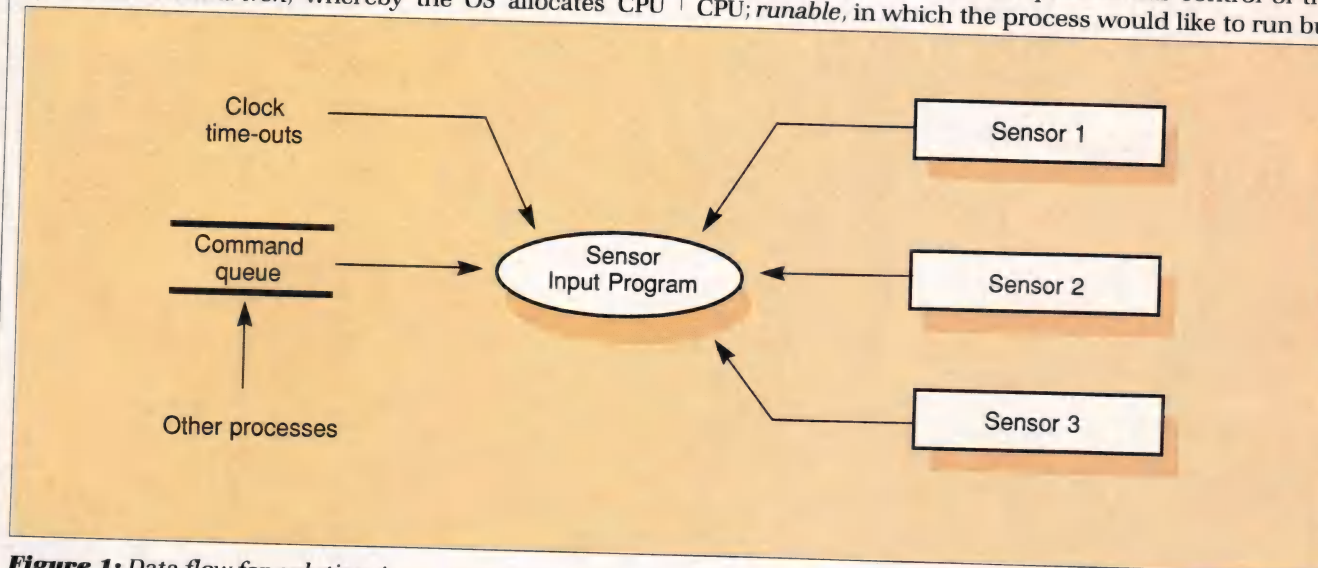


Figure 1: Data flow for solution A

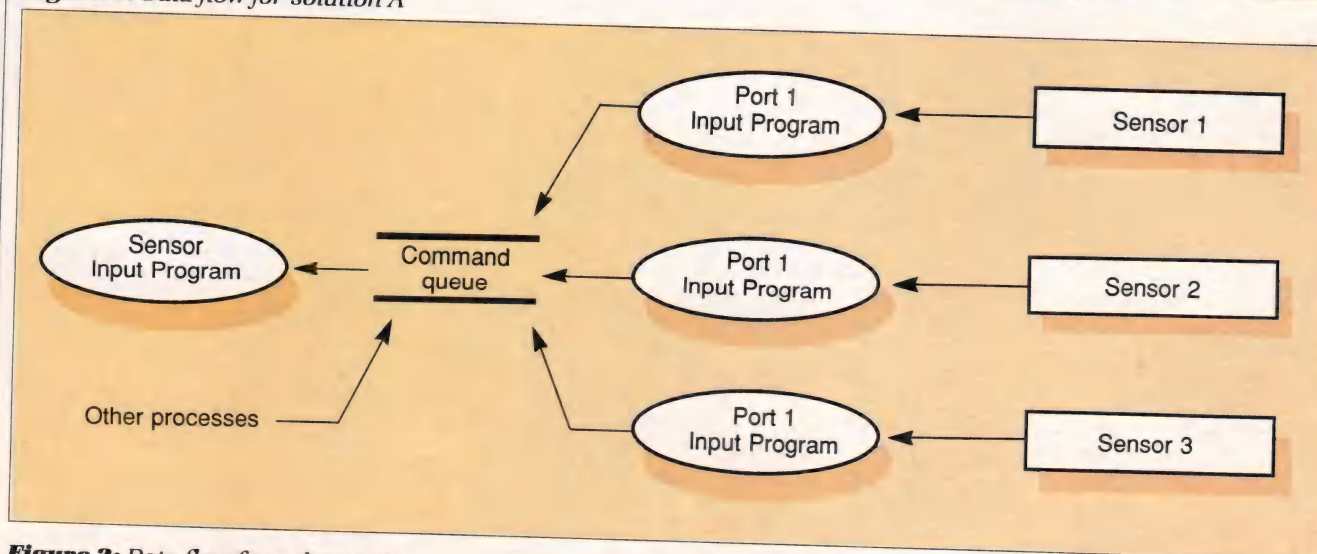
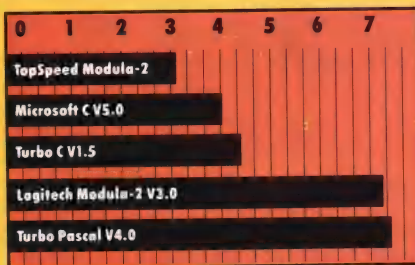


Figure 2: Data flow for solution B

FACT:

THE COMPILER THAT MAKES THE FASTEST CODE FOR SIEVE IS JPI TOPSPEED™ MODULA-2.

The successor of Pascal:
JPI TopSpeed Modula-2
produces better code than
Microsoft C, Turbo C,
Logitech Modula, and
Turbo Pascal 4.0. The fig-
ures speak for themselves:



Measured by British Standards Institution (BSI)
(25 iterations of Sieve on 8MHz AT)

In England and Europe contact:

Jensen & Partners UK Ltd., 63 Clerkenwell Rd.,
London EC1M 5NP, Phone: (01) 253-4333.
In England call Toll Free 0800 444-143,
24 Hours. Compiler Kit £59.95, TechKit
£29.95 (introductory offer only valid in the US).



Jensen &
Partners
International

TopSpeed is a trademark of Jensen & Partners International.
Other brand and product names are trademarks or registered
trademarks of their respective holders.

CIRCLE NO. 139 ON READER SERVICE CARD

JPI TopSpeed Modula-2 is a professional
Modula-2 development system with full
support of memory models, multi-tasking,
long data types, structured constants, long
and short pointers, 80×87 inline code and
emulator, separate compilation, direct
BIOS/DOS calls etc. The comprehensive
library includes CGA, EGA and VGA
graphics support, math functions, sorting,
file handling, window management and
more. Here is what our users say:

*"JPI Modula-2 is the Modula-2
we have all been waiting
for. JPI Modula-2 will do for
Modula-2 what Turbo Pascal
did for Pascal."*

—K N King
Author of Modula-2:
A Complete Guide

*"JPI Modula-2 is a landmark
product. The environment is
better than anything on offer
from Borland or Microsoft."*

—Huw Collingbourne
Computer Shopper

The Compiler Kit includes: High-speed optim-
izing compiler, integrated menu-driven
environment with multi-window/multi-file
editor, automatic *make*, fast smart linker.
All Modula-2 sources to libraries included.
Bonus: Complete high-speed window man-
agement module included with source.

The TechKit includes: Assembler start-up
source code for system, JPI TopSpeed
Assembler, TSR module, communications
drivers, PROM locator and technical
information.

Systems requirements: IBM PC or compat-
ible, 384K available RAM and two floppy
drives (hard disk is recommended).

To become part of the excitement, make
use of our limited-time introductory price
of \$59.95 (after July 4, 1988: \$99.95).
30 days unconditional money-
back guarantee.

To Order:

Call 1-800-443-0100
Ext 255, 24 Hours.
Or mail in the coupon
or a letter.

Yes, I want to own the compiler that
makes the best code. Please rush me:

☐ Compiler Kit \$59.95

Please add \$5 shipping and handling.

☐ TechKit \$49.95

SEND TO: Jensen & Partners International,
1101 San Antonio Rd., Suite 301, Mountain View CA 94043, Phone: (415) 967-3200

Name: _____

Address: _____

City/State/Zip: _____

Credit Card # _____

Exp. Date _____

☐ Visa ☐ MC ☐ Check

Disk Type ☐ 5 1/4" ☐ 3 1/2"

CA residents add sales tax.



Our workstations are en those with a passion for p

While some companies sell a lot of computers because they make something for everyone, we sell a lot because we don't.

All the workstations we make, the applications that run on them, and the networking power that unites them with the other computers in your company were created for a select group of people.

Namely the engineers, product designers, software developers and other professionals who demand nothing less than ultimate compute performance.

People who clamor for access to processing power and graphics. Who possess an insatiable

appetite for information. And who can ill afford to endure the delays, limitations and obstacles that typically hinder the effort to attain it.

If you're such a person, you should have an Apollo workstation. For you'll realize the moment its screen is in front of you that the issue of performance is behind you.

An Apollo workstation will grant upon you enough dedicated compute power to keep your imagination charged permanently. Letting you choose from a compatible family of workstation systems whose prices start as low as a personal computer and whose perfor-



gineered for performance.

mance extends to that of supercomputers.

These machines will grant you imagery so brilliant you won't want to blink for fear of missing something. With real time two- and three-dimensional graphics that render up to 16.7 million colors at 130,000 vectors per

second. And they'll open your eyes even wider with networking power and elegance.

Every Apollo workstation, from the Series 3000™ Personal Workstation™ to our new Personal Supercomputer,™ functions as a command center from which you have unequalled access to data, processing power, development tools, and applications.

So that every mainframe, minisuper, and microcomputer on your network is at your beck and call.

In a manner almost invisible to you, our workstations show you networking performance you probably thought impossible.

For with the industry's first implementation of Network Computing Architecture,™ they make your multi-vendor network appear as one computing environment.

Letting you run a single application on a network of computers by automatically dispatching portions of a program to the processors most qualified to execute them. And providing the tools to develop and debug code running on different machines.

All while freeing you to create applications, access network resources and even move from one operating environment to another with whatever language, menus and file names you define.

A fact that might inspire you to wonder if we don't engineer our workstations only for you.

Today, there is more than one way to measure computer performance. But when the criteria include processing power, graphics and network computing, nothing measures up to Apollo.



apollo

For more information, call 1-800-323-1846. In Massachusetts call 1-800-847-1011. Or write Apollo, 330 Billerica Road, Chelmsford, MA 01824.
Apollo is a registered trademark and Series 3000, Personal Workstation, Personal Supercomputer and Network Computing Architecture are trademarks of Apollo Computer Inc.

some other process currently has control of the CPU; and *blocked*, in which the process is waiting on some event—a clock interval, operator input, output to flush, and so on.

Unix, like many other systems, makes most of its OS requests *synchronously*, and they become blocked until the system can service them. Most real-time systems, on the other hand, make systems calls that may lead to blocking *asynchronously*, whereby the calls make their request for the resource but continue normal processing. When the resource becomes available, the OS notifies the process in one of two ways—either with an *event flag* or through a *completion routine*. Many real-time processes use a combination of event flags and completion routines.

Event Flags

Event flags are semaphores that indicate whether or not a condition has become true—for example, a process may ask the system to read a device and set the event flag when it has finished reading in the data. A real-time process may have several outstanding asynchronous system requests pending—the program may make a request to read input from a port, it may schedule a clock time-out, and it may request a read from an interprocess queue. Real-time operating systems allow the process to become blocked until one flag or a combination of flags become set.

Unix has a similar concept, called *semaphores*. The most common use of semaphores, however, is to provide locking on some shared resource such as a shared memory table. Unix provides no way to link a *read()* request to a semaphore, for example; hence it provides no way for a process to become blocked while waiting for the system to service multiple requests.

Completion Routines

Completion routines are called by the OS on the behalf of a process when the OS has finished with some requested function. (In some environments completion routines are also known as *asynchronous system traps*, or *asynchronous system routines*.) Generally, you can think of completion routines as the software equivalent of hardware interrupts.

If the OS finds that another event has become true while the process is still handling the original completion routine, it can either handle that event right away or else queue up the routine and handle it after finishing with the current completion routine. Some real-time systems also allow prioritized event handling, like CPUs that prioritize hardware interrupts.

Most systems allow a program to pass some predefined data to a completion routine; hence the program may have a single completion routine servicing similar requests. You may want to set up read requests on several sensors using the same completion routine for each; when the system asynchronously calls the completion routine, it will pass some argument block identifying the particular sensor that has been read.

Unix has no concept of completion routines, and because all resource requests are synchronous, it has no need to notify a process that a particular request has been completed, either through an event flag or by calling a completion routine. (Unix allows you to input calls that return immediately if no data is present. Although this doesn't provide an asynchronous interrupting ability, it does allow a program to set up its own polling loop without becoming blocked waiting on a single event. One of the examples I discuss later uses this sort of "no wait" input.)

Unix does, however, have a construct called a *signal*, which behaves similarly to a completion routine in that when a process receives a signal (from the operating system or from another process), the signal will begin execution of some predefined routine. Signals were designed to handle asynchronous hardware exceptions such as bus, floating-point, and addressing errors as well as user abort requests and program time-outs.

Once a process has intercepted a signal, the expected response is usually to perform some cleanup (close files, release resources, maybe print an error message) and then exit. By default Unix assigns a set of routines to handle this cleanup; however, a program also has the option of setting up its own functions that Unix will call when it receives a particular signal.

The signal-handler function can perform any operation. Because of the nature of the operating system, however, while in the handler code, the process is in a very fragile state. For example, after Unix calls the signal-catcher routine, it automatically resets the signal catcher to its own default value. If the process receives a second signal before the handler has had a chance to reset the signal catcher, Unix will call its own routine, which may abort the program.

The only signal type that you can use successfully as part of your normal programming is the clock time-out signal, *SIGALRM*, which is a special type of signal that you can schedule from within your program via the *alarm()* system call. Although you cannot predict exactly when your program will receive the *SIGALRM* signal, you can keep track of whether or not your program has issued the *alarm()* call; hence your program can take some precautions to handle the interruption. One of the sample programs accompanying this article illustrates a "safe" *SIGALRM* handler.

Two Alternatives

When building a real-time Unix process or porting an existing process from a real-time environment, you can choose two basic plans of attack. The first involves simulating the real-time environment by setting up a scheduler routine within your process. Although your scheduler won't be as time efficient as a real-time OS scheduler, this method does have the advantage that the Unix version will have the same basic structure as the real-time version. This is particularly attractive when porting an existing process to Unix.

The second method of building a real-time process requires that you understand Unix's limitations in the real-time environment and structure your process so that it takes advantage of Unix's strengths. As mentioned

Now
Supports
Quick
BASIC 4.0,
Turbo
Pascal
4.0

Programmers: Go home early!

C, BASIC, Pascal, dBASE®, FORTRAN and Modula-2 programmers:
be more productive by clarifying and documenting your source code.

"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."

— PC Magazine

Source Print and Tree Diagrammer both have easy-to-use menus with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Join thousands of programmers who are working more efficiently using Source Print and Tree Diagrammer. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee. **Order both and save. Only \$155.00.**

800-257-5773 Dept. F6

MasterCard, VISA, American Express, COD. Add \$5 for shipping/handling.

or see your local dealer!

Source Print and Tree Diagrammer are trademarks of Powerline, Inc. dBASE is a trademark of Ashton Tate. Prices subject to change without notice.

Source Print™

organizes your source code, simplifies debugging, and makes documentation a snap! It lists one or more source files with informative page headings and optional line numbers, while offering invaluable features:

The Index (Cross-Reference list) saves you time by showing exactly where variables are used and where functions, procedures, and routines are called.

\$97⁰⁰

Locations where new values may be assigned to variables are shown, making it easy to track down that mysterious value change.

Structure Outlining solves the problem of hard-to-see nested control structures by automatically drawing lines around them.

Automatic Indentation of source code and listings reduces your editing time and ensures indentation accuracy.

Plus... Source Print generates a table of contents listing functions and procedures. Keywords can be printed in boldface on most printers. Multi-statement BASIC lines can be split for readability. Functions and procedures can be drawn by name from one or more source files to form a new file.

Tree Diagrammer™

shows your program's overall organization at a glance. Ordinary program listings merely display functions, procedures, and subroutines sequentially, but do not display the relationships between these routines. Our revolutionary new Tree Diagrammer automatically creates an "organization chart" of your program showing the hierarchy of calls to functions, procedures, and subroutines. Recursive calls are indicated and designated comments in the source code will appear on the chart.

Tree Diagrammer helps you organize your program more logically. And you'll be amazed at how easy it is to debug when you see how your routines interact.

\$77⁰⁰

Before

Before

```
150 FOR INDEX = 1 TO 100
160 IF TB(INDEX) = 0 THEN X = 5
170 C = 50: WHILE K <= 1000: TB(K) = 0: K = K + X: WEND
180 GOSUB 2000
190 XT(C) = X
200 T2(C) = K
210 C = C + 1
220 NEXT INDEX
```

BASIC

After

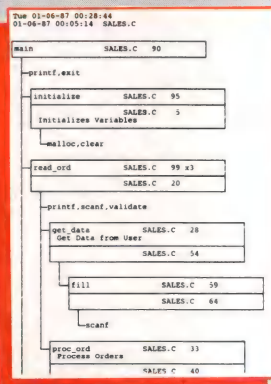
Wed 12-31-86 07:22:03 INDEX (Cross Ref)

all identifiers				
inrecord	4.191 21.889 23.990	9=396 22.922	19.825 22.953	19=826 23=978
ins	53.2293 54.2331 54.2354	53=2309 54.2332 54.2364	53=2319 54.2336 54.2365	53.2325 54=2346 54.2366
intext	4.193 43=1820	9=395 45=1902	43.1796	43.1815

Index

```
1 PUBLIC value, val1, val2, val3
2 USE VALUE IN FROM DATA
3 date=CONCAT("12/30/87")
4 DO WHILE detail < "end" (12/30/87)
5   IF NOT "Enter detail" = GET DA
6     READ
7   ENDIF
8   value = 0.00
9   val1 = 0.00
10  val2 = 0.00
11  val3 = 0.00
12  IF PRASED < detail
13    DO CASE
14      DO CASE
15        CASE Selector = "1"
16          DO PROC
17            CASE Selector = "1"
18              value = value + Qua
19              val1 = val1 + Qua
20              val2 = val2 + Qua
21            ENDIF
22          CASE Selector = "2"
23            IF Qua < 0.00
24              value = value - Qua
25              val1 = val1 - Qua
26              val2 = val2 - Qua
27            CASE Selector = "3"
28              value = value + Qua
29              val1 = val1 + Qua
30              val2 = val2 + Qua
31            ENDIF
32          ENDIF
33        ENDIF
34      ENDIF
35    ENDIF
36  ENDIF
37  IF "value" =
38    DO CLEARUP
39    CLEAR ALL
```

dBASE



Powerline, Inc. 2531 Baker Street, San Francisco, CA 94123 415-346-8325

YES! Rush me ☐ Source Print @ \$97. ☐ Tree Diagrammer @ \$77. ☐ Both \$155. Ship/Handling \$5. For CA add 6% tax ☐ Total

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

☐ Check enclosed ☐ VISA ☐ MasterCard ☐ American Express

Card # _____ Exp. Date _____

Signature _____ Phone # _____

F6

earlier, Unix has no real notion of asynchronous system resource requests nor of the real-time constructs used for implementing them (event flags and completion routines). The key, therefore, is to write your processes so that they don't require asynchronous system requests.

That last statement seems intuitively obvious and may seem intuitively impossible as well. With a little forethought, however, you can create a process that will run with nearly the same efficiency as a version running on a dedicated real-time operating system.

The Problem

Suppose, for example, that you have a process whose primary duty is to read a set of sensors and process the data. If the process doesn't receive any input from the sensors after some time interval, it should alert some other process. In the background it may also need to deal with commands coming from other processes. Figure 1, page 20, shows the basic data flows.

For the sake of simplicity, let's assume that the sensors are connected via a standard serial port and that sensor data comes in over the link in new-line-terminated ASCII strings. The algorithm and code omit the actual input processing as this isn't important to the example. The program listings also omit error checking; for any production program, you will, of course, need a generous portion of error checking.

Solution A—Overriding the Unix Scheduler

One method for solving the problem is to make Unix believe that it is a real-time OS. This solution is inferior to Solution B (described later), but for some applications, particularly for quick-and-dirty ports from real-time systems, you may prefer it.

Listings One-Three, pages 50-55, show the solution written in C. Notice that the main loop attempts to read each of the sensors as well as the command queue. If *read()* finds data present, it will process the data; otherwise, it will continue polling the other inputs. To prevent the process from hogging the CPU in an endless loop, it will delay between loop iterations.

This program includes two functions—*nap()* and *marktime()*—that are worthy of further discussion. *Nap()* provides a program delay of a finer granularity than Unix normally provides with the *sleep()* system call. *Marktime()* provides a consistent interface to the *SIGALRM* clock signal.

Nap()

The Unix system call *sleep()* delays a program with a granularity of 1 second. Most versions of Unix actually implement this by delaying the program until the next second boundary after the indicated time rather than an exact number of seconds following the point at which your program calls *sleep()*. Hence, *sleep(1)* may delay your process anywhere from a single clock tick to nearly a full second after invocation. In most instances, this is acceptable. For the sample program, however,

The Custom 386 Programmer's Workstation

Looking for a lightning-quick 386 system that's tailored to your needs? CAE/SAR Systems, Inc. will custom-fit you a 386 system more powerful than most on the market. Whether it's a system designed for your program development, artificial intelligence, CAE, or systems design work, CAE/SAR delivers reliable, powerful 386 workstations built for today's programmers.

Based on a proven 386 motherboard, CAE/SAR 386 systems come in dozens of different configurations for memory, disks, floating point and graphics. You can select high speed drives (16 ms), 70Mb, 140Mb, or 300 Mb; EGA or mono monitors and cards; and 2.5 Mb, 4.5Mb, or 8.5Mb 32-bit RAM—plus other options!

The CAE/SAR 386 systems run Unix and DOS concurrently, and also run OS/2

and Xenix. Floating point options are available for the Intel 387 chip.

Basic Unix/Xenix systems start at \$3,495.

Get a system that fits you perfectly. Call CAE/SAR Systems today for more information.

CAE/SAR Systems, Inc.

P.O. Box 50243
Palo Alto, CA 94303
(415) 949-3816

Genuine
25 MHz
machines
available now!

FREE

Disk cache package
and Turbo EGA for
hot performance
in disk and
graphics!



CONVERT YOUR PC, XT OR AT INTO A HIGHER FORM OF LIFE!

386 MOTHERBOARDS FOR 386 SPEED

Don't let your PC give up the ghost — Hauppauge has just arrived with a new spark of life: the 386 MotherBoard.™ Far more advanced than an accelerator card, our line of MotherBoards grace your PC, PC/XT, PC/AT or compatible with speeds equal to the IBM PS2 Model 80. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket for speeds that make users humble with awe.

OS/2 Compatible. To ensure a long, fruitful life, our 386 MotherBoard is compatible with the PC/AT (BIOS and I/O) — so you can run the new generation of DOS, OS/2. Our Board also runs Windows/386, UNIX V and PC-MOS/386. For more power, you'll find 16-bit expansion slots that accommodate the latest I/O expansion cards. No 386 accelerator card gives you so much versatility. **Only our 386 MotherBoard gives your PC a future with unlimited possibilities!**

The Critics Applaud! *PC Magazine* awarded our Board "The Editor's Choice" for 386 Replacement Boards. *PC World* called it "the Upgrade Product of the Year."

Technical Features ■ 16 MHz 80386 ■ 1 Megabyte of 100 nsec 4-way interleaved RAM ■ PC/AT compatible I/O and BIOS for support of OS/2 ■ Six 8-bit expansion slots ■ two 16-bit expansion slots (four on 386 MotherBoard/AT) ■ One 32-bit expansion slot for up to 12 Megabytes of high speed memory ■ Battery-powered clock/calendar

386 MotherBoard/PC or MotherBoard/XT	\$1495
386 MotherBoard/AT	\$1595
32-bit RAM Board	
(2 Mbytes installed; up to 4 Mbytes)	\$795
16 MHz 80387 math coprocessor	\$695
16-bit combination hard disk/ floppy disk controller	\$245

For more information on our easy-to-install MotherBoard, call:
1 (800) 443-6284. In New York, call **(516) 434-1600.**

Hauppauge Computer Works, Inc.
175 Commerce Drive,
Hauppauge, New York 11788

Hauppauge!

REAL-TIME

(continued from page 26)

let's arbitrarily decide that you need to poll the input sensors every 300 milliseconds by using the *nap()* function as shown in Listing Two.

Nap() works by using a special characteristic of the standard terminal driver. Normally, a terminal driver reads characters from a port until it sees a carriage return and then returns the input string to the process. This is known as *canonical processing*. Canonical processing also understands user-defined ASCII characters for backspace and line erase. Canonical processing is appropriate for reading input when the user is typing in input from a terminal.

Via an *ioctl()* system call, you can disable canonical processing and set two input parameters—*VMIN* and *VTIME*—so that a *read()* from a port will return after either reading *VMIN* characters or else delaying *VTIME* ticks (expressed as tenths of a second). *Nap()* uses (some may say abuses) this characteristic by attempting to read from an unused port. Naturally, it will find no characters available and will return a failure after a delay of *VTIME*.

Marktime()

I mentioned earlier that the Unix signal provides a function similar to a completion routine. The *alarm()* system function call allows a program to schedule the signal *SIGALRM* sometime in the future, and the *signal()* system function call allows the program to define a

signal handler. By themselves, *alarm()* and *signal()* provide a limited means of implementing a completion routine. They are limited in that only one alarm can be outstanding at any given time and that they have no implicit means of passing data to the completion routine. Also, a program that needs to set alarms from different parts of the program has no means of coordinating the different time-outs except through the use of global data—a poor programming practice.

Marktime() and its associated functions form a shell around *alarm()* and *signal()*. They allow the calling program to set up multiple time-outs; each time-out can have its own completion routine, a flag that will be set when the time-out expires, and a pointer to a unique data block. The calling process can also disable, enable, and cancel time-outs through the functions associated with *marktime()*. Note in Listing One that the program disables time-outs when not explicitly processing the main loop; this ensures that the time-out won't unexpectedly interrupt other processing.

Listing Three shows *marktime()* and its associated functions.

Solution B—Restructuring the Problem

For most situations, the preferred method of solving the problem is to work within the limits of Unix. Remember that Unix processes can only block on one system call at a time, typically a *read()*, a *pause()*, or a *sleep()*.

Instead of trying to make a single Unix process handle all the possible events, as I did in solution A, you can reorganize the main routine into several small proc-

Sierra™ OPS5

Create sophisticated expert systems on your PC with the expert systems language.

Sierra OPS5 is a fast, sophisticated, absolutely 100% complete implementation of OPS5 designed specifically for the PC.

FULL: We left nothing out — 32 bit integers, real numbers, files, 'build', external functions, ... complete!

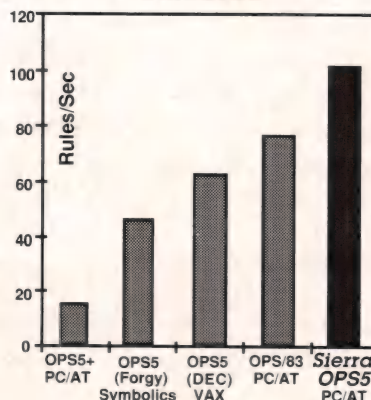
FAST: Running the same program used to benchmark other expert system languages, Sierra OPS5 topped them all. No other data-driven production system language, interpreted or compiled, matched our performance!

FLEXIBLE: *DEVELOP* your expert systems using the fully integrated workbench environment — multiple windows, multi-buffer editor, incremental compilation, full tracing and debugging. *COMPILE* your knowledge base to create a stand-alone executable OPS5 program. *EMBED* multiple independent knowledge bases in your own 'C' programs and call them when and as often as YOU want!

RESOURCE EFFICIENT: The runtime libraries require as little as 40K!

Requires an IBM PC/XT/AT or compatible with 384K RAM, DOS 2.0 or later, Microsoft C V4.0, V5.0, or QuickC required for external functions or stand-alone program. Trademarks: OPS5+/Computer* Thought Corp., OPS/83/Production Systems Tech., IBM/ International Business Machines, Microsoft/ Microsoft Corp., Symbolics/Symbolics Inc., DEC/Digital Equip. Corp.

NASA
"Monkeys & Bananas"
Benchmark



**Inference
Engine
Technologies**

PERSONAL VERSION \$129.00

- 250 rules
- OPS5 Workbench
- Compiler
- Runtime Libraries
- External Functions
- Full Manual & OPS5 textbook
- Non-Commercial License

COMMERCIAL VERSION \$495.00

- Personal Version plus:
- 1000 - 1500 rules
- Full Embedability within 'C' programs
- Commercial Royalty-free runtime license
- 1 year of updates
- Telephone support

Shipping & Handling \$5.00, \$20.00 outside US and Canada. Visa/MC/POs/Drafts on U.S. banks. MA residents add 5% sales tax.

1-800-255-0625
in MA 923-0998

1430 Mass. Ave., Suite 306-I
Cambridge, MA 02138

IMMEDIATE DELIVERY!
1-800-265-4555

List Price: \$495
Introductory
Price:
\$295*



Exp. Date _____ Signature _____
WATCOM and Express C are trademarks of WATCOM Systems Inc.
© Copyright 1988 WATCOM Products Inc.

Introducing DESQviewTM 2.0 API Tools

DESQview API Reference Manual

This is the primary source of information about the DESQview API. It contains all you need to know to write Assembly Language programs that take full advantage of DESQview's capabilities. The Reference manual comes with an 'include' file containing symbols and macros to aid you in development. AVAILABLE NOW!

DESQview API C Library

The DESQview API C Library provides C Language interfaces for the entire set of API functions. It supports the Lattice C, Metaware C, Microsoft C, and Turbo C compilers for all memory models. Included with the C Library package is a copy of the API Reference Manual and source code for the library. AVAILABLE NOW!

DESQview API Debugger

The DESQview API Debugger is an interactive tool that enables the API programmer to trace and single step through API calls from several concurrently running DESQview-specific programs. Trace information is reported symbolically along with the program counter, registers, and stack at the time of the call. Trace conditions can be specified so that only those calls of interest are reported. AVAILABLE JUNE 88!

DESQview API Panel Designer

The DESQview API Panel Designer is an interactive tool to aid you in designing windows, menus, help screens, error messages, and forms. It includes an editor that lets you construct an image of your panel using simple commands to enter, edit, copy, and move text, as well as draw lines and

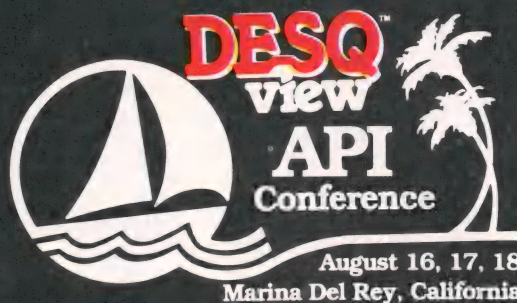
Bringing new power to DOS

boxes. You can then define the characteristics of the window that will contain the panel, such as its position, size, and title. Finally, you can specify the locations and types of fields in the panel.

The Panel Designer automatically generates all the DESQview API data streams necessary to display and take input from your panel. These data streams may be grouped together into panel libraries and stored on disk or as part of your program. AVAILABLE JUNE 88!

DESQview API Pulldown Menu Manager

The DESQview API Pulldown Menu Manager is an interactive tool to aid you in designing pulldown menus. This DESQview API tool assists you in giving your DOS program an OS/2-like look and feel. AVAILABLE JULY 88!



Call for registration information (213) 392-9851

Quarterdeck
Quarterdeck Office Systems
150 Pico Boulevard
Santa Monica CA 90405
(213) 392-9851

MS-DOS and IBM PC-DOS are trademarks of Microsoft Corporation and IBM Corporation respectively.

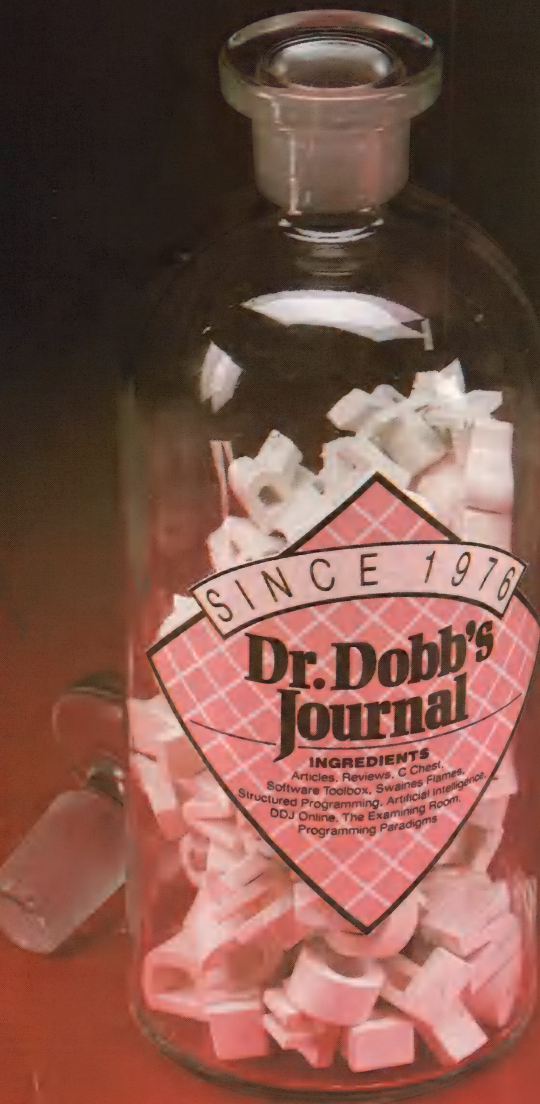
CIRCLE NO. 204 ON READER SERVICE CARD

THE CURE FOR COMMON CODE

Are you getting the recommended monthly allowance of C, Assembly, Forth, Pascal, BASIC or Modula-2? Subscribe to *Dr. Dobb's Journal of Software Tools* and you won't catch any nasty bugs again!

Each month the Doctor brings you aid for ailing algorithms and the cure for common code. For the latest developments in software design and pages of code that will make you a more productive programmer, take the Dr. Dobb's prescription.

For more than a decade, the programming elite have known *Dr. Dobb's Journal* to be the foremost source of software tools. Subscribe now and get your monthly dose from the Doctor.



PASCAL
FOR
BASIC
MODULA-2
ASSEMBLY
C

Dr. Dobb's Journal of Software Tools

The
R_x
for
Programmers

Subscribe
Now &

Save
Over
15%

Off the
Newsstand
Price!

SUBSCRIBE AND SAVE!
Subscribe to

DR. DOBB'S JOURNAL OF SOFTWARE TOOLS
and save over \$5—a 15% savings off the cover price!

☐ 1 year \$29.97 ☐ 2 years \$56.97

☐ Please charge my:

☐ Visa

☐ Master Card

☐ American Express

☐ Payment enclosed

☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

ONLY \$29.97! YOU SAVE OVER \$5.00

*Savings based on the full one-year newsstand rate of \$35.40. All foreign countries please add \$13 per year for surface mail; Canada & Mexico add \$29 per year for airmail; other countries add \$32 per year for airmail. All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank. Please allow 6-8 weeks for delivery.

A Publication of M & T Publishing, Inc.

375S

SUBSCRIBE AND SAVE!
Subscribe to

DR. DOBB'S JOURNAL OF SOFTWARE TOOLS
and save over \$5—a 15% savings off the cover price!

☐ 1 year \$29.97 ☐ 2 years \$56.97

☐ Please charge my:

☐ Visa

☐ Master Card

☐ American Express

☐ Payment enclosed

☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

ONLY \$29.97! YOU SAVE OVER \$5.00

*Savings based on the full one-year newsstand rate of \$35.40. All foreign countries please add \$13 per year for surface mail; Canada & Mexico add \$29 per year for airmail; other countries add \$32 per year for airmail. All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank. Please allow 6-8 weeks for delivery.

A Publication of M & T Publishing, Inc.

375S

COMMENTS & SUGGESTIONS

Dear Reader,

Dr. Dobb's has a long tradition of listening to its readers. We like to hear when something really helps or, for that matter, bothers you. In this hectic world of ours, however, it is often difficult to take time to write a letter. This card provides you with an easy way to correspond and, if you include your name and address, we may use appropriate comments in The Letters column. Simply fill it out and drop it in the mail. —Ed

Which articles or departments did you enjoy the most this month? Why?

Comments or suggestions _____

Name: _____

Address: _____



BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools

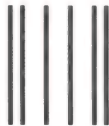
Box 3713
Escondido, CA 92025-9843



No Postage
Necessary
If Mailed
In The
United States



**Dr.
Dobb's
Journal
of
Software
Tools**



BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools

Box 3713
Escondido, CA 92025-9843



No Postage
Necessary
If Mailed
In The
United States



**The
R_x
for
Programmers**

**Subscribe
Now &**

**Save
Over
15%**

**Off the
Newsstand
Price!**

PLACE
STAMP
HERE

Dr. Dobb's Journal of
Software Tools

501 Galveston Drive
Redwood City, CA 94063

REAL-TIME

(continued from page 28)

esses. Each of the small processes will be responsible for a single blockable resource; when one of the processes has completed its duty (for example, when a sensor input process has read a block of data), it will pass a message back to the main process via an IPC message queue. The main process will then have only one blocking state, which is a *msgrcv()* of its message queue. Figure 2, page 20, shows the data flows for this solution.

Although this method incurs a degree of overhead in the form of message passing, it simplifies the problem by removing the polling/sleeping loop, and hence the overall performance will generally be better than that in solution A.

Listings Four-Six, pages 61-65, show the C code for this solution. Note that the listings don't show how the system starts the smaller processes; in a real application the main process may *fork()* and *exec()* the smaller processes as children. Also, note that the child processes have no means of relating exception conditions to the parent process; one way to do this would be to define additional messages.

Finally, note that the child processes break the *read()* using a *marktime()* time-out. An alternative to this would be to use noncanonical input from the port using the *VTIME* and *VMIN* parameters to control the time-out. The down side to using this method is that the terminal driver no longer parses according to new-line delimiters, which means that the program must handle its own parsing of the input stream. Although this method may provide a cleaner interface, particularly if the input is a binary stream rather than ASCII, I used *marktime()* to illustrate its use with a pseudoevent flag.

Conclusion

Both solutions have merit under the right circumstances, and both can successfully perform the required chores. The best solution to running a near-real-time process under Unix, however, lies in understanding the nature of the operating system—its strengths and weaknesses—and working with the system instead of against it.

Although the solution requires some extra nontraditional planning (nontraditional in the sense of how you might build the product under a real-time operating system), restructuring the problem as I did in solution B provides the most efficient and trouble-free product.

Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call 415-366-3600, ext. 221. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

(Listings begin on page 50.)

Vote for your favorite feature/article.
Circle Reader Service No. 1.

Quaid Analyzer Instruction Display

```
dx ax      0000 0000
ds:si bx 86c4:003e 085d
es:di cx 86c4:0000 0a9a
ss:sp bp 86c4:0946 00a2
data      09c2:0008
code      09c2:0419
cs:ip      09c2:0419
          ....oditsz.a.p.c
flags 0000001001000110
```

```
d28c 0419 >move dx,ss
cc8b 041b move cx,sp
fa 041d cli
c88c 041e move ax,cs
d08e 0420 move ss,ax
0d60bc 0422 move sp,0d60
0200c481 0425 add sp,0200
fb 0429 sti
52 042a push dx
51 042b push cx
53 042c push bx
51 042d push cx
30b4 042e move ah,30
21cd 0430 int DOScall
```

Part of a Quaid Analyzer display

Quaid Analyzer is a powerful diagnostic tool that shows what is going on inside your computer. The > at the top is the cursor. You can move it with the arrow keys. When you move the cursor off the screen, the instructions scroll like text in an editor. You can move the cursor into a register and change its value, or see the instructions or data it points to. Of course, you can scroll through the data display as well, and type new values into memory. With **Quaid Analyzer** you never have to type a command.

This example shows the first instructions executed when *VDISK.SYS* installs itself. You can see that it changes stack pointers, then gets the DOS version number. We got to this point by loading **Quaid Analyzer** before DOS, then watching the DOS call and disk interrupts until the driver was loaded, then putting a breakpoint on its first instruction. Drivers are installed before DOS gives you the first prompt. What other software tool can show you a device driver install?

Quaid Analyzer comes with a manual, and software on a 3 inch and a 5 inch diskette. If you are not satisfied with **Quaid Analyzer**, you can return it within 30 days for a refund. **Quaid Analyzer** is not sold by dealers in the United States or Canada. It is not copy-protected.

To order **Quaid Analyzer**, call us with your credit card, or send us a check for \$200 US funds. We ship within a day at our expense.



Quaid Software Limited
Third Floor Dept D633
45 Charles Street East
Toronto Ontario Canada M4Y 1S2
(416) 961-8243

Warning! For advanced programmers only.

MESSAGE-PASSING OPERATING SYSTEMS

High performance and flexibility in a realtime, multitasking, multiuser, or networked environment.

by Dan Hildebrand

Operating systems in real-time environments must be capable of handling the demanding intertask communications problems that are inherent in communications, process control, and other real-time applications. At the same time, the operating system (OS) must be capable of providing the functional capabilities of a traditional multiuser OS while delivering fully network-distributed processing and the deterministic performance of a real-time executive. One way to achieve both performance and flexibility is message passing architecture.

The performance and flexibility of a message-passing OS enable the data flow on the network to consist of intertask messages. Tasks can communicate with any other task, anywhere on the network. The network then functions as a homogeneous, tightly connected array of computers, rather than a collection of computing islands connected on a func-

tionally limited network.

Conventional wisdom would have us believe that the 8088 and 80286 possess a "flawed" architecture, leaving them unsuitable for multitasking. Contrary to popular opinion, the design of these processors are admirably suited to multitasking. It is only the "flawed" architecture of conventional operating systems that negatively impact their performance. Operating system design is the primary limiting factor in the multitasking performance of these processors.

The Dilemma of the Layered Approach

In a conventional OS, various unrelated pieces of the OS often share common code and data space for convenience of implementation. Software layers over existing facilities (the "yet-another-layer" design philosophy) provide additional OS function ability. With each new release, this ever-increasing depth of layering results in progressively worsening performance in the following three crucial areas:

- The synchronization overhead incurred when a task communicates a

request to the operating system.

- The transparency of intertask communications.
- The efficiency of intertask communications.

For example, to provide network services, a layer is often added to catch OS requests and reroute them through the network software/hardware to a file server. The file server is then running a network control task that interfaces the network requests to the local OS. This "network-services" layering imposes a performance penalty on all network transactions.

To avoid performance losses, extensions can be coded into the kernel itself, thereby having access to data structures and code fragments not necessarily needed for the extension. However, these pathological connections result in side effects that can be difficult to debug and maintain. You face the dilemma of choosing between:

- 1) extending the complexity of the OS kernel at the expense of reliability and maintainability; or
- 2) extending the OS services through the addition of multiple, performance-robbing layers around the existing OS.

Dan Hildebrand is a programmer for Quantum Software Systems Ltd. (Ottawa, Ontario) and the developer of Qterm, a high-level terminal emulation program.

SideStepping the Dilemma—The Message Passing Solution

One technique that solves the performance difficulties of intertask communication is message passing. This technique involves the copying of a block of data (the message) by the OS kernel from the data space of one task to that of another. Whether the tasks are executing on the same processor or on physically remote processors does not matter. Obviously, this approach is particularly effective in integrated-network, distributed, and parallel processing environments.

An important characteristic of this approach is that message data must be physically copied from the source task to the destination task. This physical copying of the message accomplishes a "disconnection" between the two tasks, thus allowing the tasks to run on different processors (if necessary). If one of the two tasks provides OS-related services, this disconnection easily results in a networked operating system.

While performance optimization techniques may encourage the passing of pointers to messages (rather than passing message contents), this optimization has negative effects. In actual practice, the time for data transfer (passing the message) does not represent a significant portion of the task-switching process. The task switch itself represents the bulk of the operation. Also, the vast majority of messages are only a few bytes long.

In specialized applications where large buffers must be passed between tasks and where networking is not an issue, a pointer to the necessary buffer can be passed within the message. If the message is not physically copied, many additional details must be managed. The primary problem is that a sending task cannot modify or release a message buffer until the receiving task has indicated that it is finished with the message. The synchronization issues that must then be addressed only complicate and impede the operation of the system.

Conventional, layered operating systems typically protect themselves

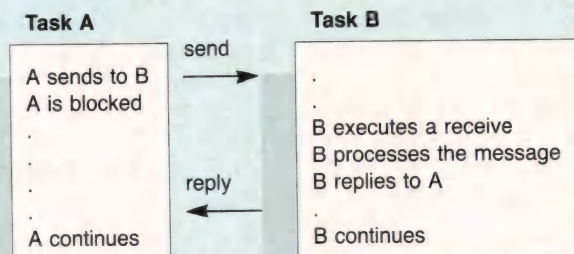
from user tasks by rigidly separating memory into "system" and "user" areas. An OS built from a group of cooperating tasks that pass messages can be set up without distinct system and user memory spaces. The only necessary system memory management is that already provided to support user tasks. A system task is then treated the same as a user task except that the system task is providing a resource intrinsic to the OS.

The dilemma confronted when expanding a conventionally structured OS is neatly sidestepped with this multiple task arrangement. Extensions to the OS are painlessly added

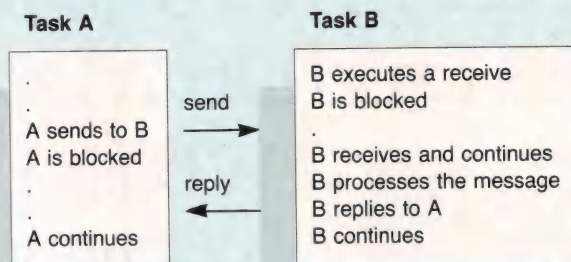
as additional tasks that efficiently pass messages to the existing OS tasks. Maintenance of the OS also can be easily managed because each task is responsible for only a well defined set of services, requested through an explicitly defined set of messages.

If memory-management hardware is available, an additional benefit of this structure is that all user and OS tasks are protected from one another. The 80286 microprocessor running in protected mode is an example of an environment within which this protection is available. The modular nature of this type of OS design is highly reliable and easily

There are two sequences that can occur when two tasks are communicating with each other using the send/receive operators. If Task A sends to Task B before Task B is ready for a receive, the following sequence occurs:



If Task B executes a receive before Task A does the send, the synchronization operates as follows:



Throughout this process there are a number of states a task can assume. These states are "send blocked," "receive blocked," and "reply blocked."

Send Blocked

The task has done a send which has not yet been received.

Receive Blocked

The task has done a receive before any other task has sent to it.

Reply Blocked

The task has done a send which was received but a reply has not yet been sent back.

Figure 1: Send/Receive Message Passing

modified and maintained.

Message-Passing Operating System Interface

The application interface to a message-passing OS is quite different from the OS interface provided by operating systems such as OS/2, PC-DOS, or Unix. Such operating systems require the application program to execute a software interrupt, or subroutine calls passing either the data for the request in the processor registers or through a pointer to a predefined table or buffer. The OS then expects to be able to directly read and write into the data space of the application making the OS request. This requires that the OS be resident on the same CPU as the task making the request and results in severe problems for networked versions of these operating systems.

Additional layers of software (which decrease overall system performance) are required to solve this problem. In contrast, a message-

passing interface produces an OS with a single, unified interface that works for communication between either local tasks or remote tasks. This unified interface results in a smaller, leaner OS that need not support two sets of interfaces. While other OS interfaces require the applications to be written in both "networked" and "non-networked" flavors, as well as requiring the operating system to support two sets of interfaces, a message-passing OS means that an application in a message-passing OS need only be written for a single interface.

One such message-passing operating system is QNX, designed and developed by Gordon Bell and Dan Dodge as an outgrowth of research done at the University of Waterloo in Canada. This operating system was introduced in 1982 by Quantum Software Systems Ltd. It is currently being used at over 55,000 sites in applications ranging from integrated office automation systems to robotics and real-time process control systems.

Although its underlying architecture is much different from Unix,

the QNX interface itself is Unix-like. The OS consists of a group of cooperating tasks that pass messages among themselves in order to accomplish various OS requests. These tasks are referred to as administrator tasks because they are essential to the operation of the OS. When an application task requires OS services (such as device I/O, task creation, and so forth), messages are sent to the administrator task that provides the required service. If those services are required of another workstation or node in the network, those same request messages need only be sent to the administrator tasks on the remote node. This message redirection is handled transparently by the system.

The kernel holds together all of the administrator tasks. The QNX kernel, which represents 10K of highly optimized code, has the primary function of performing the message-passing and task synchronization functions within the OS. A task scheduler that has set priorities within the kernel provides QNX with the deterministic response time necessary for real-time applications. On an 8-MHz 80286, the kernel performs 3,200 task switches per second; on a 16-MHz 80386, 7,200 task switches per second. Assuming another interrupt is not being serviced, a worst-case interrupt latency of 30 microseconds is experienced on an 8-MHz 80286.

System Administrator Tasks

Various administrator tasks are placed around the kernel. "Task" is the task which provides facilities relating to task creation, task death, memory allocation, and task name registration. These are given the highest priority in the system. The protected-mode 80286 version of QNX supports 150 tasks, while the real-mode 80286 or 8088 versions support 64 tasks.

Fsys is the task that implements the QNX file system. It manages the on-disk data structures that represent files and directories. Messages can be sent to this task to request operations related to the file system (such as file opening, closing, reading and writing, as well as absolute disk block manipulation). Fsys im-

"This is the
programmer's editor
I wish I'd had when
I wrote the
Norton
Utilities."

**THE NORTON
EDITOR**

- Created to meet the needs of programmers.
- Lightning fast—one of the fastest editors in the MS-DOS world.
- Easily customized and saved.
- Split-screen editing.
- Condensed/Outline display.
- Structured programming features, including Auto-indenting.
- Word features for writing documentation.
- Supports mouse.
- From the people who brought you the Norton Utilities and Norton On-Line Programmer's Guides.

Designed for the IBM® PC, PC-AT and DOS compatibles. Available at most software dealers, or direct from Peter Norton Computing, Inc., 2210 Wilshire Blvd. #186, Santa Monica, CA 90403. To order: 800-451-0303 Ext. 40 (Visa and MasterCard welcome). 213-453-2361. MCI Mail: PNCL. Fax 213-453-6398. ©1987 Peter Norton Computing.

Peter Norton
COMPUTING



QNX vs. OS/2 UNIX

QNX: Bend it, shape it, any way you want it.

ARCHITECTURE If the micro world were not so varied, QNX would not be so successful. After all, it is the operating system which enhances or limits the potential capabilities of applications. QNX owes its success (over 55,000 systems sold since 1982) to the tremendous power and flexibility provided by its modular architecture.

Based on message-passing, QNX is radically more innovative than UNIX or OS/2. Written by a small team of dedicated designers, it provides a fully integrated multi-user, multi-tasking, networked operating system in a lean 148K. By comparison, both OS/2 and UNIX, written by many hands, are huge and cumbersome. Both are examples of a monolithic operating system design fashionable over 20 years ago.

MULTI-USER OS/2 is multi-tasking but NOT multi-user. For OS/2, this inherent deficiency is a serious handicap for ter-

minal and remote access. QNX is both multi-tasking AND multi-user, allowing up to 32 terminals and modems to connect to any computer.

INTEGRATED NETWORKING Neither UNIX nor OS/2 can provide integrated networking. With truly distributed processing and resource sharing, QNX makes all resources (processors, disks, printers and modems anywhere on the network) available to any user. Systems may be single computers, or, by simply adding micros without changes to user software, they can grow to large transparent multi-processor environments. QNX is the main-frame you build micro by micro.

PC's, AT's and PS/2's OS/2 and UNIX severely restrict hardware that can be used: you must replace all your PC's with AT's. In contrast, QNX runs superbly on PC's and literally soars on AT's and PS/2's. You can

run your unmodified QNX applications on any mix of machines, either standalone or in a QNX local area network, in real mode on PC's or in protected mode on AT's. Only QNX lets you run multi-user/multi-tasking with networking on all classes of machines.

REAL TIME QNX real-time performance leaves both OS/2 and UNIX wallowing at the gate. In fact, QNX is in use at thousands of real-time sites, right now.

DOS SUPPORT QNX allows you to run PC-DOS applications as single-user tasks, for both PC's and AT's in real or protected mode. With OS/2, 128K of the DOS memory is consumed to enable this facility. Within QNX protected mode, a full 640K can be used for PC-DOS.

ANY WAY YOU WANT IT QNX has the power and flexibility you need. Call for details and a demo disk.

THE ONLY MULTI-USER, MULTI-TASKING, NETWORKING, REAL-TIME OPERATING SYSTEM FOR THE IBM PC, AT, PS/2, THE HP VECTRA, AND COMPATIBLES.

Multi-User	10 (32) serial terminals per PC (AT).	C Compiler	Standard Kernighan and Ritchie.
Multi-Tasking	64 (150) tasks per PC (AT).	Flexibility	Single PC, networked PC's, single PC with terminals, networked PC's with terminals. No central servers. Full sharing of disks, devices and CPU's.
Networking	2.5 Megabit token passing. 255 PC's and/or AT's per network. 10,000 tasks per network. Thousands of users per network.	PC-DOS	PC-DOS runs as a QNX task.
Real Time	2,800 task switches/sec (AT).	Cost	From US \$450. Runtime pricing available.
Message Passing	Fast intertask communication between tasks on any machine.		

For further information or a free demonstration diskette, please telephone (613) 591-0931.

Quantum Software Systems Ltd. • Kanata South Business Park • 175 Terrence Matthews Crescent • Kanata, Ontario, Canada • K2M 1W8

UNIX is a registered trademark of AT & T Bell Labs. IBM, PC, AT, XT and PS.2, PC-DOS and OS/2 are trademarks of International Business Machines. HP and Vectra are registered trademarks of Hewlett-Packard Company.

CIRCLE NO. 203 ON READER SERVICE CARD

plements a tree-structured file system that supports disks up to 1 Terabyte (a million Mbyte) in size with a space-efficient 512-byte unit of allocation. This file system supports random seeks within files from any point to any other point with a single, direct disk seek. Unlike typical file systems, intervening disk blocks need not be read to perform large seeks. This means QNX can be used for large, multiuser database applications. Because the file system is also power-fail safe, QNX is also

suitable for harsh environments. File ownership, attribute and permission checking usually found within a multiuser file system is also handled by Fsys. Block-oriented device drivers can be installed using a "mount" command and become an extension of the Fsys task. See accompanying text about "Mounting Device Drivers." Special tasks can also be written to adopt a drive for special purposes.

Dev is the task that performs character-oriented I/O. Drivers for the console, serial, and parallel devices are present within this task. Additional drivers can be mounted as back-

ground tasks, which can then adopt device names from the Dev task for special applications. The drivers within this task perform all the handling for options (such as flow control, line editing, baud rate changes, and so forth). Changes to option settings are performed by utilities that send the appropriate messages to Dev, thus commanding Dev to modify the requested options. Also present are library routines that allow user programs to communicate these requests. A set of routines that implement high-speed video output are included in Dev. Since these routines are integrated into the terminal independent screen and keyboard library, programs can be written that perform instantaneous screen updates on the console, while retaining terminal independence for terminal or modem applications. On a PC AT, 19 physical devices are supported, in addition to the 40 virtual device names available for adoption by device-driver tasks. The fast task switching and low interrupt latency of QNX allow many more serial devices to be supported than under conventional, non-real-time, Unix derived operating systems.

Idle is a null task executed whenever all the other tasks in the system are in a blocked state and waiting for an external event either to occur or to complete. Idle runs at the lowest priority in the system.

Net is the task within QNX that performs message passing between machines on a network. This task exists only in networked versions of QNX and occupies approximately 20K of memory. (By comparison, the standard networking extension for PC-DOS is nearly 190K in size.)

The user-extendable Timer is an optional task that can be started by the user to add complex timing capabilities. Other tasks can request "timer" to provide timeouts ranging from one millisecond to many years. Because of the real-time scheduling within QNX, tasks can be accurately scheduled with very precise timing resolution.

The Queue manager is a task that can perform queued message passing similar to that provided in mailbox-oriented operating systems. The standard send/receive, intertask message calls within QNX are blocking.

BRIEF™ Users: Now you can have fast compilation AND an integrated, productive environment.

Introducing
BRIEF v2.1

Over 5,000 of you were forced to make sacrifices to use BRIEF, The Programmer's Editor. Advanced compilers and new programming environments, like Turbo C and QuickBASIC, took up so much RAM that BRIEF could not fit in the same 640k.

If you wanted to retain BRIEF's uniquely powerful features¹ while working with larger programs, you had to sacrifice speed and continuity. Instead of a tight Edit-Compile-Edit loop, you had to slog through an obsolete Edit-Exit-Compile-Exit-Edit loop.

Now you no longer have to make that sacrifice.

You can enjoy the features¹ that have made BRIEF the best-selling and the best regarded² programmer's editor without sacrificing environment integration.

Version 2.1 of BRIEF can be swapped in and out with a single keystroke — allowing immediate compilation with even the largest compilers: Microsoft C5.0, QuickC, Turbo C, Lattice C, dBLX, FoxBASE+ v2.0, Clipper, etc.

¹ For example: real multi-level Undo (not simply Undelete), flexible windowing, unlimited file size, unlimited number of simultaneous files, automatic language sensitive indentation.

² For example:

"The quintessential programmer's editor." — *Dr. Dobb's Journal* "Right out of the box, it's a versatile, extremely powerful editor that handles most any programming task with aplomb." — *Computer Language* "Simple to learn and use and extremely sophisticated. Strongly recommended." — *PC Magazine* "Not only the best programmer's text editor I've ever seen, but it is also a tour de force in the way it was conceived and implemented." — *Computerworld* "So far surpasses users' expectations that it is revolutionary." — *MicroTimes Magazine* "BRIEF is truly outstanding." — *Microsoft Systems Journal*

Current BRIEF Users:

Call Ann for details on 4 other important enhancements. Registered users of versions 2.0 or 2.01 update for only \$35.

Call toll-free today
800-821-2492

**Solution
Systems**

Haven't tried BRIEF yet?

BRIEF retails for \$195. Call Ann today for a no-risk, 60-day trial with a full, money-back guarantee.

Call toll-free today
800-821-2492

541 Main Street, Suite 410
South Weymouth, MA 02190

The Ada® world just changed. Again.

Meridian continues its tradition of Ada "firsts" by introducing a powerful new set of integrated software development tools.

AdaVantage v2.1 Optimizing Ada Compiler

Available for the IBM PC and compatibles and the Apple Macintosh. An exceptionally fast validated Ada compiler that for the first time brings a true world-class production quality Ada compiler to your desktop personal workstation.

AdaVantage Debugger An interactive source-level debugger for use with programs written using the Meridian AdaVantage compiler. The debugger allows the programmer complete control over the execution of an Ada program in high-level Ada terms — no knowledge of the underlying machine architecture is required. The debugger supports breakpoints, subprogram traces, single-stepping, call backtraces, and full Ada reference syntax.

Ada Developer Interface A powerful interactive screen-oriented interface to the Ada library dependency information. Includes built-in operations to edit, "pretty print", compile, and link any Ada library unit. Configuration file allows user tailoring of all operations and displays.

Run-Time Customization Library For preparation of Ada application programs for execution on 80x86-based embedded systems or other operating systems. The Library is a collection of Ada source files, batch files, and documentation that

define the customizable, system-dependent components of the AdaVantage Run-Time System.

AdaStarter Identical to the validated v2.1 AdaVantage compiler with limitations that permit up to ten library units, each with up to two-hundred executable statements (unlimited declarations and comments). This is approximately equivalent to a 4000 line program. The price of AdaStarter is applicable towards purchase of the AdaVantage production compiler. Get the full power of Ada for only \$99!

AdaDesigner A collection of tools supporting the design, programming, and documentation phases of the software life cycle. Tools include a structured editor/synthesizer coupled to a text editor/incremental syntax analyzer for Ada, an incremental editor for design languages, a program derivation processor that guarantees your design is always in sync with your implementation, and a structured documentation generator.

Configuration The compilers all run in a standard PC configuration with 640K of memory (1MB on the Macintosh) and a hard disk.

For more information call 1-800-221-2522.
In California, call 714-380-9800.

“ **Meridian Software Systems AdaVantage version 2.0 compiler is an Ada software milestone.** ”
— PC Week

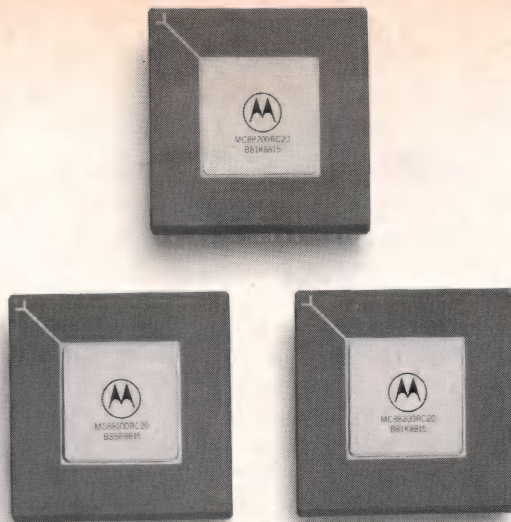


23141 VERDUGO DRIVE • SUITE 105 • LAGUNA HILLS, CALIFORNIA 92653
800/221-2522 (Outside California) • 714/380-9800 (Inside California)
Telex: 650-268-0547 MCI • Fax: 714/380-1683

□ The information contained herein is subject to change without notice. □ Ada is a registered trademark of the U.S. Government (AJPO). AdaVantage, AdaTraining, AdaDesigner and AdaStarter are trademarks of Meridian Software Systems, Inc. References to other computer systems use trademarks owned by the respective manufacturers. □ Copyright ©1988 Meridian Software Systems, Inc. All rights reserved.

CIRCLE NO. 154 ON READER SERVICE CARD

FROM
NOW ON,
THERE'S
ONLY ONE
RISC*
WORTH
TAKING.



INTRODUCING THE MOTOROLA 88000 MICROPROCESSOR FAMILY: THE GREATEST RISC OF ALL.

The future of RISC computing has been reduced to three small, but amazingly powerful chips.

Namely, the Motorola 88000 family.

One awesome microprocessor unit, supported by two cache memory management units. Designed to take RISC architecture far beyond anything else in the marketplace.

The 88000 runs at a blistering 14-17 MIPS, 7 million floating point operations per second, and an incredible 50 MIPS in parallel processing applications (using just four 88000 chip sets on our HYPERmodule™ card).

Which makes everything from multi-user business systems to fault tolerant on-line transaction processing systems to artificial intelligence systems several times faster and more powerful than ever before.

What's more, it comes with absolutely every bit of hardware and software needed to build your system of the future, today. In fact, many leading hardware and software companies, including those in the independent consortium 88open, are already designing systems around the 88000. And many more will follow.

So make sure your future is as rewarding as it can possibly be. Call us for more information at 1-800-441-2447. Or write Motorola Inc., P.O. Box 20912, Phoenix, AZ 85036.

Because the greater the RISC, the greater the reward.



MOTOROLA

**Reduced Instruction Set Computer*

© 1988, Motorola Inc.
HYPERmodule is a trademark of Motorola Inc.

MESSAGE-PASSING (continued from page 38)

Unless a conditional receive has been explicitly requested, these calls do not allow the sending task to continue until the message has been received. This blocking design is deliberate within the operating system, although it may not be convenient for some system designs. To support those designs that require it, the Queue manager task can be started to provide network-wide, queued message passing. Unlike the OS/2 Queue manager, this Queue manager buffers entire messages,

rather than just pointers to messages. This allows queues to be used across the network (if necessary). If performance optimization is necessary and network transparency is not important, the message stored in the queues can be a pointer to the message.

Through a request to Task, user-written tasks are able to become admin tasks. As admin tasks, they share the same privileges as the original set of tasks that make up the OS. Being able to start admin tasks allows the initial functional capabilities of the OS to be extended at run-time.

An essential characteristic of an admin task is that it cannot be arbitrarily killed by other tasks. Typically, admin tasks are commanded to shut down and to release any system resources the tasks may have allocated. An admin task is also able to detect the death of other system tasks so that resources allocated by the admin task for those dead tasks may be released.

Send/Receive Message Passing Primitives

QNX implements two message passing primitives—send and receive. These primitives are unbuffered,

OS/2's A Real-time Alternative

Neither Microsoft nor IBM touts OS/2 as a real-time operating system. Nevertheless, programmers might write OS/2 applications that must track real time. This is particularly true when programmers are developing communications applications to monitor events and take action when responses fail to occur as expected. Real-time tracking can provide the user with a specific time period in which to perform some action, or perform an action such as saving an editor's buffer on a regular, timed interval. Real-time control can also allow an application to run at preset time intervals. OS/2 has several timer service functions to facilitate writing real-time control routines.

The only fly in the OS/2 real-time control ointment is OS/2's main feature—multitasking. Because multiple threads and processes can be running simultaneously, real-time tracking that uses the CPU clock can never be totally accurate because a higher-priority process may be eating up CPU cycles. But multitasking has its advantages, too. Using multiple threads, a program can synchronize several different hardware devices so that they can perform simultaneous tasks. Timers can likewise synchronize the activities of several asynchronous programs.

OS/2 provides both synchronous and asynchronous timer services. DosSleep is the synchronous func-

tion that puts your application on hold so that you do not need delay loops. DosTimerAsync, DosTimerStart, and DosTimerStop are asynchronous functions that allow you to start, stop, and read software timers, using system semaphores to alert an application when timing functions have finished executing. The timer starts when it is called and then control passes back to the calling thread, which resumes execution. The thread and the timer execute concurrently. Upon completion of the timer's interval, the timer clears a semaphore. The calling thread can check the semaphore to see if timing is complete. For example, if your program issues a DosTimerAsync(5000, mysem, semidentifier) call, a timer with a five-second interval begins execution and at the end of the interval clears the semaphore mysem, which you can read on the file handle semidentifier. Your program must create and set the semaphore by using the DosCreateSem and DosSemSet functions before you call the asynchronous timer.

DosTimerStart operates much the same as DosTimerAsync but continues to run while clearing its associated semaphore each time the timer interval elapses. You must reset the semaphore after it is cleared. DosTimerStop is used to halt DosTimerStart. The DosSleep function acts as a synchronous timer. The

thread that calls DosSleep suspends its execution for the interval of the timer. A DosSleep(5000) call puts its calling thread on hold for five seconds.

If your programs merely need to synchronize the flow of data among threads or processes, semaphores and shared memory enable you to exploit one of several OS/2 communication paths between processes. Particularly within a single monolithic application, the private semaphore/shared memory interprocess communication technique has speed advantages over the nonprivate (but slower) pipes mechanism for sharing data. Pipes pass data only between parent processes and their children. Two-way communication between such processes requires two separate pipes.

The periodic clock interrupt (or timer tick) of OS/2 occurs 32 times each second. This means that timing functions carry a 1/32-second quantization error. Therefore, you should think in term of seconds (not milliseconds) when using timers of OS/2. High-precision timing of events happening in the millisecond range should use other methods to measure time intervals. For example, you can make repeated DosGetDateTime calls to read the Date/Time date structure's contents into a buffer or into variables for computing the passage of small time intervals. Another solution is to check

blocking operations that cause the task issuing a send request to be blocked if the target task is not correspondingly receive-blocked. When two tasks are in complementary send/receive states, the message is transferred and the receive task becomes unblocked (see Figure 1, page 35). The highest-priority task will then run. QNX always executes the highest priority, unblocked task. If two tasks are compute bound at the same priority level, round robin task scheduling will occur. Should an event occur that causes a higher priority task to become unblocked, QNX will pre-empt the currently exe-

cuting task, and switch to the higher priority task. Fast, pre-emptive task scheduling is essential to real-time applications.

An important aspect of the send/receive operations is that time-ordered queuing is performed whenever more than one task attempts to communicate with the target task. Multiple send requests to a single task performing a receive are queued in the order they were received and are processed in sequence. The target task has the option of completely servicing the first request before servicing the next request (or additional requests).

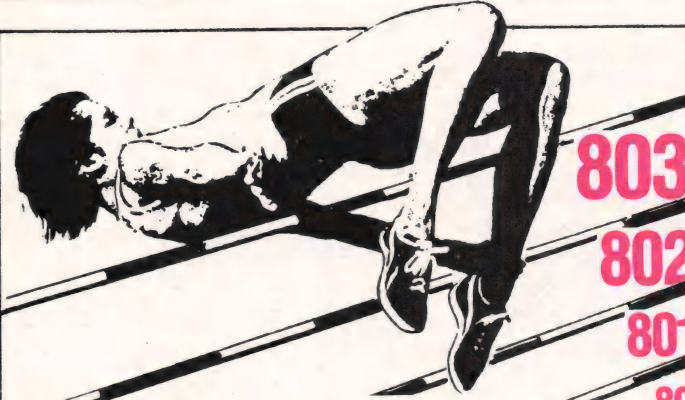
In addition to the send/receive operators, mechanisms called exceptions, ports, and registered names are available for intertask communication. These additional mechanisms are useful for special cases where send/receive communication is not appropriate.

An exception is similar to the signal found in Unix. An exception is an asynchronous event that can cause an exception handler within the task to be executed in response to the exception. Exceptions are valuable because they can be used to break out of the send/receive blocked states. The most common exception

the time bytes in the read-only global information segment by using the DosGetInfoSeg call.

You can alleviate the problem of timer service threads losing clock cycles to higher priority threads by elevating the priority of the thread making the timer calls. If you make these threads the highest-priority threads, you will ensure that events needing critical timer servicing won't lose clocks to higher-priority threads. The major difficulty developers face in writing real-time control software is interrupt processing. OS/2 does not allow an application to process hardware or software interrupts. You can process interrupts only with OS/2 device driver. Since these drivers are very difficult to write in a high-level language, creating routines to handle interrupts—which can then notify and pass data to an application—will be a complex process. In short, OS/2 should be considered a workable real-time operating system alternative only for those custom applications where you are in total control of the environment and can therefore, worst case interrupt latency and task scheduling times. —G.M.V.

G. Michael Vose, co-editor of the newsletter "OS Report: News and Views on OS/2." He can be reached at Box 3160, Peterborough, NH 03458.



80386
80286
80186
8086
8088

Scale new heights with OASYS ...

Our Intel Software Tool Kit keeps on growing ...

TOOL KIT

- Fully optimizing C, Pascal and FORTRAN Green Hills Compilers with full 80386 Support
- Microsoft Cross C and MASM
- 80386 Macro Assembler
- 80386 Linker with Intel, COFF, and MS object formats
- Symbolic Debuggers
- Full complement of compilers, assemblers, linkers, debuggers and simulators for 80286/8086

FEATURES

- 80386 compilers utilize multiple optimization techniques producing extremely compact code
- Tool Kit provides complete support for 80387 and Weitek 1167 Floating Point Units
- Complete Run-Time Library support
- 80386 Assembler accepts Intel/Microsoft instruction mnemonics
- Produces COFF, Intel-compatible, and MS Object modules
- Assembler/Linker backwards compatible with entire 8086 family, including 80286/8086
- Softprobe II 8086/186 Simulator and Debugger
- Microsoft Cross C assures complete MS-DOS compatibility

AVAILABILITY

Native on 80386, MS-DOS and UNIX
For Cross-Development Targeting 80386/286/86:
VAX; Sun; IBM PC; Apollo, NEC; ...Others being ported

You name it ...

We are a "Single Source" supplier for more than 120 products running on, and/or targeting the most popular 32-, 16- and 8-bit micros and operating systems.

We Specialize In:
Cross/Native Compilers C, Pascal, FORTRAN, Modula-2, Prolog, C++ — Assemblers/Linkers — Symbolic Debuggers — Simulators — Interpreters — Profilers — QA Tools — Design Tools — Comm. Tools — OS Kernels — Editors — PC Attached Processors and more

We Support:
680xx, 80x86, 320xx; CLIPPER, and dozens more

We Provide:
Porting and technical assistance — OEM arrangements — Custom development — Let us be your external tools group

Oasys

(617) 890-7889 230 Second Avenue
P.O. Box 8990, Waltham, MA 02251-8990

Trademarks are acknowledged to: Intel Corp., AT&T, DEC, Weitek Corp., Phar Lap Software, Microsoft Corp., IBM, Systems & Software Inc., Apollo Computer Inc., Sun Microsystems, Inc., and XEL, Inc.

CIRCLE NO. 180 ON READER SERVICE CARD

Mounting QNX Device Drivers

Within QNX, device drivers can be started or mounted at run time and do not need to patch nor be compiled into the kernel of the operating system. Drivers for block-oriented devices (hard disks, floppy disks, RAM disks, and so on) can be written as specially structured interrupt handlers which are connected as particular drive numbers (using the mount command) to the Fsys (file system) task. This type of driver becomes a directly callable set of routines within the code space of the Fsys task which are able to do low-level block read/write operations while the Fsys task manages the file system data structures within those blocks.

Mounted Fsys drivers are automatically used by the Fsys task when access to the appropriately numbered drive is requested. Since the Fsys task is network accessible like any other task, the drivers attached to Fsys are accessible using standard file system calls (*fopen*,

fclose, and so forth).

A mechanism to adopt a disk drive also exists, allowing any foreign disk format to be easily supported with a driver task for that format. From an application's point of view, when the *fopen* message for a given drive is sent to the Fsys task that owns the drive, a special reply message will be returned, indicating which task has adopted the drive. The *fopen* library routine will then resend the same *fopen* message to the task on the node that has adopted the drive. From this point on, all further requests for that drive will be routed directly to the task that has adopted the drive.

This facility is used to advantage by the DFS task (Dos File System). When started, this task can bring into the device list any PC-DOS hard disks or floppy disks, anywhere in the network. Once mounted, any QNX program that accesses files in the QNX file system can also access any PC-DOS file system and have

the file structure information translated "on the fly" for any request. For example, with the DFS task running, the QNX full screen editor can edit any file on any disk (QNX or PC-DOS) without regard for the underlying structure of the file system.

Using this technique, it is possible to write a file system administrator for a WORM drive. This drive could then be read and written by any of the standard QNX utilities. It would be up to the WORM file system task to maintain a file system on the optical drive that could handle the write-once limitation of the drive.

Drivers for character-oriented devices can be written as a task that adopts a new or existing device name so that additional devices can be accessed like any other serial or parallel device using the standard file/device calls (*fopen*, *fclose*, and so on). The implementation of a spool device which collects output in a disk file rather than printing it is an example of such a driver.

INDUSTRIAL STRENGTH OOPS.

You have three options in today's world; lead, follow or get out of the way. You've already taken a leadership position in hardware with the latest 286 or 386 system. Now you can use that triple-digit architecture to blast ahead of the pack with the most powerful new Object Oriented Programming (OOPS) software on the market: Smalltalk/V286.

Smalltalk/V, the original OOPS tool for the PC, gave scientists, engineers, programmers and educators a brand new way to solve problems. And soon they were developing exciting new applications in everything from economics to medicine to space.

Now Smalltalk/V286 gives you true work station performance with industrial strength capabilities like: push-button debugging; multi-processing; portability

between DOS, OS/2 and Presentation Manager operating environments; integrated color graphics; a rich class library; and access to 16 MB of protected mode memory, even under DOS.

The new Smalltalk/V286, which is even easier to learn and use than Smalltalk/V, retails for just \$199.95. Or you can buy Smalltalk/V, still the world's best selling OOPS, for only \$99.95. And both come with our 60 day money-back guarantee.

Check out the new Smalltalk/V286 at your dealer. If he doesn't have it, order toll free, 1-800-922-8255. Or write to: Digitalk, Inc., 9841 Airport Blvd., Los Angeles, CA 90045.

And let us put you ahead of the power curve.

Smalltalk/V286

MESSAGE-PASSING (continued from page 43)

Device drivers can also be written in the form of background tasks that implement their own message-passing interface. Complex devices such as MAP (Manufacturers Automation Protocol) interface cards do not naturally interface as block or character-oriented devices and should be implemented as a standalone control task. The network wide intertask messaging of the network then makes the service provided by the driver task a network-accessible resource.

Writing a driver for a network card that is connected to a network separate from the QNX network produces a gateway. The machine on the QNX network that contains both network cards would then run the gateway task. This gateway task could then register its name on the network, becoming a network accessible resource that can allow any QNX task throughout a network to access the remote network via the gateway task. —D.H.

is the break exception that is generated from the keyboard.

The primary use of a port is for interrupt handlers to communicate with a task. This facility makes it possible for a task to be written which contains the interrupt handler within the body of the task itself. Using standard systems calls, the task is able to attach to a port and then connect the handler to the appropriate interrupt vector. After any other internal initialization, the task can receive block itself upon the port and optionally open itself for message reception from other tasks. An "attach" or "detach" operator is used by a task to obtain a port.

During interrupt service time, the interrupt handler is able to make use of the code and data of the task. If an event requiring handling by the task or the OS results, the handler can signal the assigned port, causing the task to become unblocked in order to perform whatever service the interrupt handler required. Note

that the interrupt handler is connected directly to the interrupt vector and that no operating system overhead is added to the interrupt service time.

For two tasks to communicate, the sending task must know the node number and task identifier (ID) of the destination task. If the sending task was responsible for starting the remote task, the sending task will know this information. If the sending task is expecting to send to a previously present task, the sending task must be able to discover the node and task ID (TID) of that task. To facilitate this, the receiving task that wishes to provide a network-accessible service can register a textual name. Tasks needing to locate that task can obtain the node and TID by using the textual name that the task would have registered.

For example, one task that typically needs to register itself is a print spooler. Once registered, any task on the network wanting to print need only inquire about the spooler task and use the standard intertask messaging to send the data to be



MESSAGE-PASSING (continued from page 45)

printed to the spooler task. Multiple spoolers could be started for any printer, anywhere in the network. Any spooler and its corresponding printer could be relocated without concern for whether tasks needing that spooler would be able to locate it.

Since the primary tasks within the operating system are assigned predefined TID numbers at boot time, remote tasks are always able to communicate directly with the primary OS tasks anywhere on the network without having to first discover their TID numbers.

Flexibility

Operating system extensions to QNX are background tasks. These extensions interface to application tasks with exactly the same messaging that applications already use to request OS services. In effect, user-started OS extensions become indistinguishable from the OS itself. These additional tasks run under

the privilege and access restrictions necessary for their intended purposes and no more. This allows the protection inherent in a protected-mode OS to also apply to user-written OS extensions.

The segmentation of the OS into a set of cooperating tasks is also memory-efficient. Portions of the OS that are not needed for every application environment can be implemented as background tasks that are started only as needed for a given application environment. For example, the QNX network administrator task automatically removes itself from the system if it detects that a network card is not present.

A further example of the flexibility of this approach is that QNX is able to support PC-DOS as a task within both the real-mode or protected-mode versions of QNX. Most PC-DOS applications (including Lotus, Sidekick, dBase, and WordPerfect) can run without difficulty.

Networking and Message-Passing Operating Systems

Message-passing operating systems

provide an alternative approach to networking. Most networked operating systems deal with file transfers or remote terminal-session communications. The data flowing on a message-passing network like QNX, is the intertask messaging itself. Operations such as command loading (file transfer), remote device I/O (terminal sessions), and general intertask communications are transparently accomplished within the context of intertask messaging.

User tasks are able to make requests of any task in the system, anywhere on the network. As a result, the network provides each task with what appears to be the resources of a parallel computer, with all network resources (processors, memory, disks, devices, and tasks) accessible as local resources.

Rather than just a simple copy of a file to a device, as with the command "copy file \$lpt," QNX allows commands to be specified in a manner that takes full advantage of all of the available network resources. The next copy command demonstrates this:

Code Base

Program dBASE Applications in "C"

dBASE Compatible

- Works with dBASE compatible products
- Code Base routines correspond to dBASE commands and functions

Source Code

- 100% Included
- Quick C, Microsoft C 5.0 or Turbo C 1.5
- DOS 2.0 through DOS 3.3 or OS/2

Multi-User

- Networks are detected and used
- Supports record and file locking

Index Files

- No maximum number
- Auto-maintenance
- Compatible with dBASE

Combine the Speed and Flexibility of "C" with the Productivity of dBASE

\$149 US

Sequiter Software Inc.

P.O. Box 5659, Station L
Edmonton, Alberta, Canada
T6C 4G1 ☎ (403) 439-8171



ATTENTION SOFTWARE DEVELOPERS - AUSTRALIA AND THE PACIFIC

Improve your productivity with software tools from the experts

Author's representatives and distributors



HALO Media Cybernetics

CompuView

POLYTRON



All major credit cards accepted.

95 Canterbury Road, Middle Park, Vic. Australia 3206
Tel: +61-3-699 9899 Telex: AA31604. Fax: +61-3-699 7501.

MkAdd 1074

KEEP UP WITH THE OS/'s

Megabytes of Memory and 32-Bit Performance for DOS.

If you thought the only way to protected mode was the big move to OS/2... We have good news! You can gain the benefits of protected mode the easy way with OS/286™ and OS/386™. These tools for C, Fortran, Pascal and assembly language programmers permit rapid conversion of existing DOS applications from "real" 8086 mode to "protected" 286 and 386 mode. They don't replace or modify DOS, but extend it to protected mode. This way you get multi-megabytes of directly addressable memory (16Mb-286, 4Gb-386) with the compiler, TSRs, device drivers, graphic routines, etc. you use today.

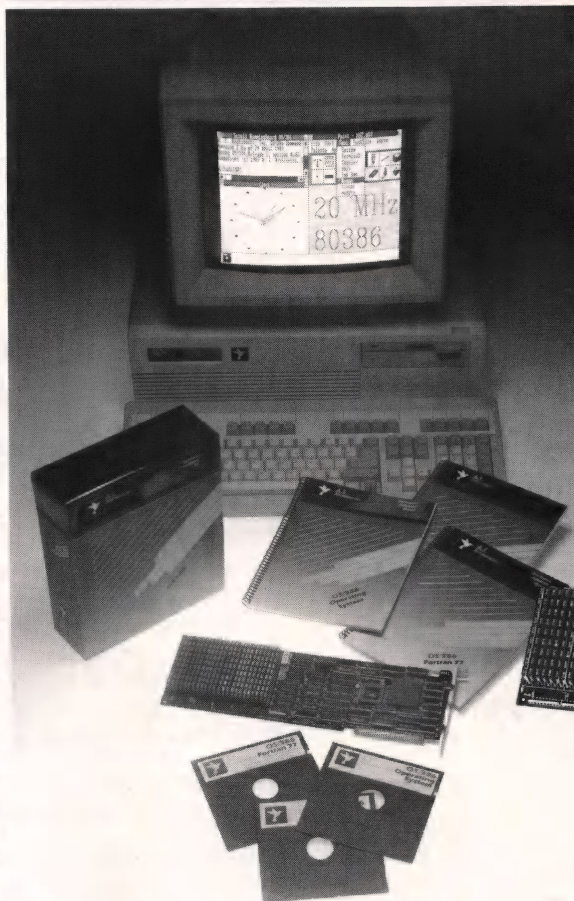
OS/286 and OS/386 are the only DOS extenders that span both the 286 and 386 processors. They run on the widest array of AT and 386 machines, with 32-bit capability today on 386s that yields twice the performance of 16-bit mode.

OS/286 and OS/386 are the preferred solutions for developers of high performance memory-intensive applications, including CADKEY, CASE, and Gold Hill, and premier language developers Lahey and MetaWare.

Our optional TOUCHDOWN™ BIOS supplement provides fast and reliable protected mode operation on *any* 286 system, even those with problems resetting the 286. (Ever notice how few existing machines Operating System/2 runs on?) TOUCHDOWN is not required for most major brand AT clones, but for the older machines it is a lifesaver!

If your applications are running out of memory or need more speed, enhance them now without abandoning your investment in DOS.

**Special
\$49.50
Evaluation Offer**



Check out for yourself the benefits of protected mode. Our \$49.50 "sampler special" is a complete OS/286 Developer's Kit, but with a time limited, non-distributable kernel. There's no better way to learn about the outstanding features of OS/286 and OS/386 than to try them. Of course, the \$49.50 is applicable to the purchase of the full OS/286 (16-bit) or OS/386 (32-bit) kit at \$495 for either one.

The OS/x86 Developer's Kits include support for popular, C, Fortran, Pascal, Lisp, Prolog, compilers, and assemblers from: MetaWare, Lahey, Microsoft, Lat-tice, Gold Hill, LogicWare and Phar Lap. A number of other packages are also supported including PLINK86, Halo & GSS Graphics and DESQview.

The HummingBoard® turns any XT, AT or 386 into the fastest, most capable 386 system available. The dual processor architecture provides up to 900K each for multiple real mode applications (Lotus, etc.) which can co-reside with protected mode applications on the HummingBoard's 20MHz 386, while

debuggers, editors, networks run concurrently on the base processor. Applications using OS/x86 automatically split tasks between the Hummingboard and the base. This can provide a 5-10x speed-up with a Hummingboard in an AT and up to a 2x performance boost with it in a fast 386 system, such as the Compaq 386/20. Hummingboards available with up to 24Mb, 80387.

**Prices start at
just \$1,495**



**A.I.
Architects, Inc.**

One Kendall Square, Cambridge, MA 02139

TEL (617) 577-8052 FAX (617) 577-9774

Credit card orders only - 24 hours. 617-577-1305

HummingBoard® is a registered trademark and OS/286, OS/386 and TOUCHDOWN are trademarks of A.I. Architects, Inc., PLINK86 is a trademark of Phoenix Corp., HALO is a registered trademark of Media Cybernetics, Inc., DESQview is a trademark of Quarterdeck Office Systems.

Compaq Deskpro 386 is a trademark of Compaq Computer Corp.

CIRCLE NO. 79 ON READER SERVICE CARD

MESSAGE-PASSING

(continued from page 46)

```
$ [2] [3] 4:/cmds/copy [1] 3:/user/peggy/  
new [4] $!pt
```

Assuming this command line were typed from the console on node 6, the command copy would be loaded from node 3, drive 4 into memory on node 2. While executing on node 2, the copy command would read data from the file new on node 1, drive 3, and send it out to the \$!pt device on node 4. If a '&' character were placed on the end of the command line, the command would execute as a background task on node 2 and freeing the user's console for other work. None of the general utilities contain special code to handle network-oriented file or device names.

In this case, the "copy" command itself is 2,700 bytes in size and operates with simple calls to standard file processing library calls.

Device access across the network is able to work in this manner because whenever a task performs an

fopen to open a file or device, it sends a message to the fsys (for files) or dev task (for devices) on the node that owns the file or device. When the requested task replies to the *fopen* request, any following read/write messages will be routed directly to the remote task that controls the device.

In a Unix environment, the network typically supports only terminal sessions and file transfers. To access a remote database, the task performing the I/O must log in to the remote node and execute the query task there. The result is the central machine (already burdened with the file and device I/O) must also execute the tasks that are generating the I/O requests for all the users on the network. A QNX network allows the tasks to execute on each user's workstation with only the remote file and device I/O requests flowing to the node (or nodes) that contain the devices being accessed.

With this naming flexibility, resources present on the entire network are part of the same "name

space" and may be operated upon just as the resources present on a single node. A program written to access files or devices by name can name any file or device on the network. The program can then have transparent access to the file or device without resorting to a special "network services" interface.

Conclusion

Through the use of a message passing architecture, QNX is able to provide real-time performance with multiuser support and full network transparency in an operating system that occupies less than 150 Kbytes, QNX can run on as limited a machine as a PC with a single floppy drive and 256K of RAM, or in protected mode on a 80286 or 80386 with 16 Mbytes of RAM. The networking transparency results in a QNX "mainframe" that can be built piece by piece to provide as much performance as necessary.

DDJ

Vote for your favorite feature/article.
Circle Reader Service No. 2.

C Programmers: Combine C and COMMON LISP to Increase the Power of Your Software

TransLISP PLUS™

Simple.

Add LISP features to your software without making it a full time job. The TransLISP PLUS tutorial, on-line help, and 30 sample programs with commented source make it easy.

Practical.

Start by modifying the LISP sample programs and including them in a system you wrote in C. Yes, in C! TransLISP PLUS includes a C Language Interface that lets you integrate your Microsoft C code and libraries with all or portions of our LISP interpreter.

Use TransLISP PLUS to add natural or command language features to replace menus... or to flexibly manage related but disparate information. Code from C libraries provided by other vendors can be integrated into your program to perform tasks not normally part of LISP.

Thorough.

TransLISP PLUS took over 400 primitives from the most widely used and respected LISP standard, COMMON LISP, and made it available on IBM PCs, XTs, ATs, and virtually every other MSDOS machine. So now you can work with anything from a \$700 PC to a \$7000 PC.

The utilities toolbox is included at no charge with a built-in editor, pretty printer, cross reference, and additional debugging tools.

An optional Runtime encrypts your source code so that you can distribute your applications safely. You pay no royalties.

Requires MSDOS 2.0+, 320K RAM, and a 360K floppy.

MONEYBACK GUARANTEE

Try TransLISP PLUS (\$195) for 30 days — if not satisfied get a full product refund. The Optional Runtime is available for \$150. Or start by learning LISP with TransLISP (\$95) then upgrade to PLUS for \$158.

Call (800) 255-4659

In MA (617) 331-0800



**The
Coder's
Source™**

541-D Main St., Suite 412, So. Weymouth, MA 02190

CIRCLE NO. 237 ON READER SERVICE CARD



The NEW Alslys Cross-Development System



Real-Time Embedded, Bare-Board Ada Now Targeted To Entire iAPX86 Family.

Efficient, compact, production quality Ada code.
Efficient, application-tailored Run Time system. On
an 8 MHz 286, a pure synchronization rendezvous
in 250 microseconds.

Real-time Ada has come of age.

The new Alslys cross-development system is now
available. And, of course, validated. It is hosted on
an IBM PC AT or compatible, and targets every
member of the Intel iAPX86 family.

The system contains two compilers and a toolset.
First, a validated Ada cross-compiler to bare 8086,
80186, 80286, 80386 and other boards. Second,
a validated self-hosted Ada compiler on the AT
that allows you to generate code for the target that
can be tested on the host.

And the tools. First, AdaProbe, Alslys' unique
debugger and program viewer for the host, and
Cross-AdaProbe for the target. And then, a complete
and uniquely powerful multi-library system, binder,
AdaReformat, AdaXref, AdaMake, downloading
utilities, configurability interface, and utilities
supporting PROM-burning.

The new cross-development system is based
on the new Alslys Version 3 "root" representing
a quantum leap forward in Ada technology. If you
thought Ada wasn't
ready for real-time,
you owe it to your-
self to call.

In the US: Alslys Inc., 1432 Main St., Waltham, MA 02154 Tel: (617) 890-0030

In the UK: Alslys Ltd., Partridge House, Newtown Rd., Henley-on-Thames, Oxon RG9 1EN Tel: 44 (491) 579090

In the rest of the world: Alslys SA, 29 Avenue de Versailles, 78170 La Celle St. Cloud, France Tel: 33 (1) 3918.12.44

Send me: ☐ New Brochure on Ada Real-Time

☐ Data Sheet on Cross-Compiling to the iAPX86

☐ New Brochure on V. 3 "Quality"

Name

Company

Address

City State Zip

Phone

Alslys, Inc. • 1432 Main Street • Waltham, MA 02154

Alslys
Version
Three
Ada
Compilers

**THE MANY FACETS
OF QUALITY**

A Simple Decompiler

Recreating source code without token resistance.

by William May

On the surface a decompiler sounds like a hackers' tool, a utility for snooping through code. Although programs such as Steve Jasik's MacNosy for the Macintosh are certainly useful, the decompiler I describe in this article has more prosaic, less imaginative uses. In my case its purpose was to speed up a product that included a spreadsheetlike component.

Fast recalculation speed is an important objective of a spreadsheet, and I felt this would best be achieved by converting the user's expressions into a form that could be computed efficiently—that is, a compiled form. Although the compilation process is straightforward, a problem arises when the user wants to see and/or change an expression: The application must be able to recreate the original expression, or "source code."

A simple way to satisfy this need would be to store both the source and the compiled expressions. This method is used in Smalltalk-80, which also provides interactive viewing and modification of compiled expressions. I felt, however, that with the limited syntax of spread-

sheet expressions, I should be able to recreate the source code from the compiled code itself.

The problem turned out to have been solved already, and the algorithm was described in an article by P.J. Brown.¹ As you will note from the publication date (1976), the algorithm was devised before the introduction of spreadsheets or of microcomputers. Brown developed the algorithm as a component of an interpreted Basic system for minicomputers, where it was used to decompile Basic statements into source form. Given its heritage, it is clear that the algorithm is capable of handling a far wider range of expressions than my spreadsheet is.

The algorithm has a few points that should make it of interest to DDJ readers. First, as I said, it can handle an extremely wide range of expressions, which allows it to find use in many applications. Second, the algorithm is driven by a table that is easy to set up and maintain. With its flexibility and simple setup, the algorithm is one of those tools that, once available, seems to find many unexpected uses. Finally, the subject of decompilers is worth exploring.

The remainder of this article covers my implementation of Brown's algorithm. The code, developed on a Macintosh in MPW C, should be portable to virtually any standard C compiler. In the remainder of this

article I assume that readers are familiar with reverse Polish notation (RPN) and have some slight familiarity with compilers.

Brown's Algorithm

Example 1, page 51, shows some examples of using Brown's algorithm, with the object code on the left and the recreated source on the right. For presentation purposes, the object code uses printable characters and standard symbols, which would not usually be the case in an application. Besides handling the normal arithmetic and logical operators, Brown's algorithm can handle Lotus 1-2-3's @ functions, BASIC's *Let...=...*; and *if...then...else...*; statements, and many others.

One important characteristic of Brown's algorithm is that it is forward scanning, meaning that it starts at the first byte of the RPN expression and works its way toward the end. The alternative, as you might guess, is backward scanning. Forward scanning has two advantages over backward scanning: the scanning algorithm can identify variable-length tokens, and the decompiler can emit (print) its results as it proceeds.

Backward scanning has the advantages of greater speed and better worst-case performance. A backward-scanning algorithm, however, produces its results in reverse, so a second pass is needed to unwind

William May works for MassMicro Systems. He is involved with Macintosh II video products and can be reached at 303A Ridgely Circle, Clinton, MA 01510.

the results and this eliminates some of the speed advantage.

A more critical drawback of backward scanning is the difficulty in handling variable-length tokens. Because variable-length tokens are the norm, this is a serious problem. A spreadsheet, for example, will typically have tokens for operators (1 or 2 bytes), cell references (varying formats for absolute addresses, relative addresses, and so on), integers (4 bytes on a Macintosh), floating-point numbers (4–12 bytes depending on type and format), strings (many formats), and so on.

At some point programmers using a source-recreation algorithm must decide how faithful the recreation will be to the original text. A normal part of the compiling process will remove all "nonessential" elements, such as spaces, tabs, line feeds, comments, and so on.

There are two solutions to the question of recreation fidelity. The first option is to attempt to achieve strict fidelity, and the second is to recreate expressions in a canonical, or standard, format. Note that the approaches are not mutually exclusive. Idiosyncratic formatting can be kept for some elements and ignored for others.

In my application—a spreadsheet—the scope for idiosyncratic formatting is limited. There are no lines, tabs, or comments; however, there is a problem with parentheses. The placement of parentheses in an expression can vary enormously from user to user. Yet this information is lost during the conversion from infix to postfix notation. Users would certainly be dissatisfied if no parentheses were recreated and would undoubtedly prefer that their original placement be maintained.

An approach to handling this difficulty is to compile information on the location of parentheses into the object code. An expression evaluator would ignore the parentheses, and Brown's algorithm would use the information to recreate their exact placement. By making a very small increase in object code size, the process of compiling and decompiling expressions becomes almost invisible to the spreadsheet user.

The Transformation Table

The core of Brown's algorithm is a transformation table that describes how operators in the input stream are to be converted back into source code. Example 2, this page, is an example of such a table. The example shows no particular "language" but illustrates several types of transformations that the algorithm can handle.

Column 1 of the table is an opcode, or identifier, symbol generated by the compiler—that is, the token the algorithm will see in the input stream. In the example I have used printable characters that correspond to their common usage—for example, + indicates addition. Generally the opcodes are encoded as 1- or 2-byte integers. Column 2 indicates the number of operands associated with the opcode. The examples show both unary and binary operators. The table could also include nonary operators (no operands) as well as more complex operators, such as *if...then...else* or *@pv(pmt, int, term)*.

The following columns, 3 through 5, describe the strings that are to be emitted by the decompiler and their position relative to the operands.

Source Recreation		
Object code		Result
abc++		a+b+c
bcdfe+*/=		let b=c/d*f+e
gabc+(*cd,, \$		let g=sum(a*(b+c),c,d)

Example 1: Source code recreation

typedef struct decomp_row {				
char	ident			
short	lex type;			
char	*prefix_1;			
char	*prefix_2;			
char	*suffix;			
} decomp_row;				
static decomp_row table[]={				
{'!',	UNARY_OP,	"-",	NIL,	NIL},
{'+',	BINARY_OP,	NIL,	"+",	NIL},
{'-',	BINARY_OP,	NIL,	"-",	NIL},
{'/',	BINARY_OP,	NIL,	"/",	NIL},
{'*',	BINARY_OP,	NIL,	"*",	NIL},
{'^',	BINARY_OP,	NIL,	"^",	NIL},
{'(',	UNARY_OP,	"(",	NIL,	"")
{'\$',	UNARY_OP,	"sum(",	NIL,	"")
{'=',	BINARY_OP,	"let ",	"=",	""
{',',	BINARY_OP,	NIL,	","	NIL}
};				

Example 2: The transformation table

For example, + is a binary operator with no *prefix_1*, a *prefix_2* (+), and no *suffix*. This means that + will apply to two operands, that there is no prefix to the first operand but there is a prefix to the second (+), and that there is no suffix. The expression *op1 op2 +* will thus be transformed into the string *op1+op2*, as expected. Another example is =, which is a binary operator with a *prefix_1* (let), a *prefix_2* (=), and a suffix (;). In this case, *op1 op2 =* is transformed into *let op1=op2;*, the classic Basic *let* statement.

Algorithm Overview

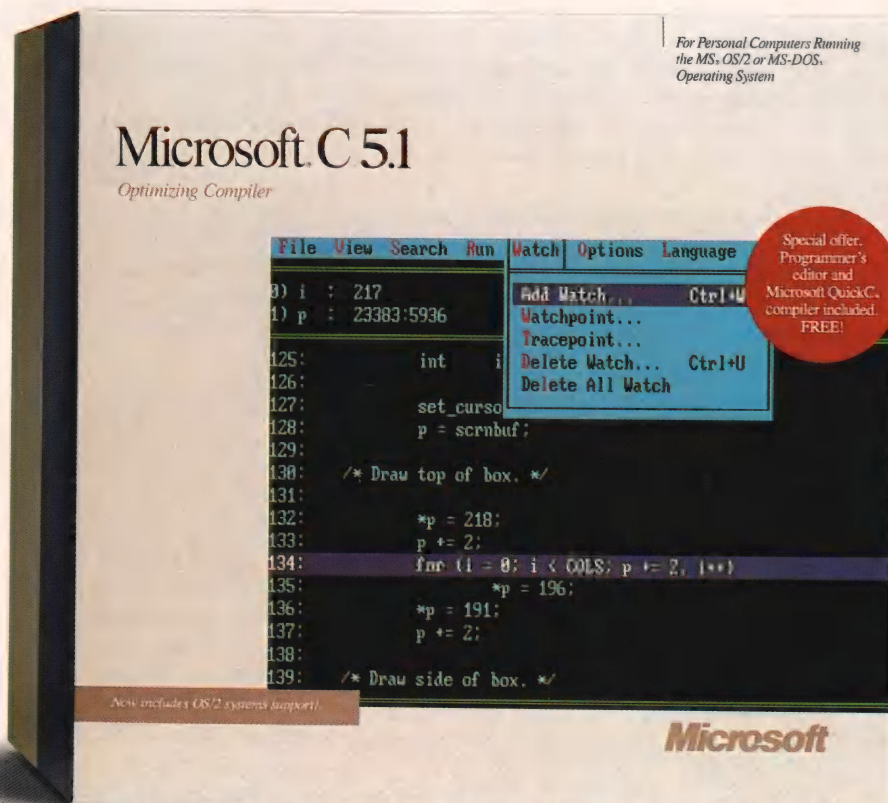
Probably the best way to get a feel for Brown's algorithm is to step through an example. The example I will follow is the expression *xy!z/=*, which will be transformed into the string *let x=-y/z;*. My example will use the table in Example 2 as its basis.

The first token found in the scan is the operand *x*. Upon seeing the operand, the algorithm scans forward to find any operators that give rise to a prefix for the operand. In this case the next token is *y*, which is also an operand, not an operator. A level count is incremented (it is now 1).

The following token, *!*, is a unary operator. According to a general rule, for an *n*-ary operator the level is decremented by *n-1*. Because *!* is a unary operator, the level is decremented by 0 (no change). Another rule is then applied, which states that if the current level is *n*, push *prefix_(-n+1)* for the operator. Because the level is 1, the algorithm should push *prefix_0*, which does not exist. Therefore nothing is pushed on the stack for this operator.

The scan continues, finding the operand *z*, which increases the level (now 2), and then the binary operator */*. Applying the decrementing rule, the level is decremented to 1. Using the prefix rule, the algorithm is once again called on to push *prefix_0*, which still does not exist. Finally, the operator *=* is seen. This is defined as a binary operator, so the level is decremented once again, to become 0. This indicates that the

Introducing the fastest possible way to create the fastest MS-DOS programs possible.



Microsoft C Optimizing Compiler 5.1 Techbox

Compiler

- Optimizations that generate the fastest code for DOS and OS/2 systems.
 - In-line code generation.
 - Loop optimizations.
 - Elimination of common subexpressions.
- Full OS/2-system support to break the 640K barrier. New.
- Family API programs that run under DOS and the OS/2 systems. New.
- Write multithreaded programs and Dynamic Link Libraries. New.
- Small, medium, compact, large, and huge memory models.
- Mix models with NEAR, FAR, and HUGE keywords.
- Fast compilation (10,000 lines/minute) with Microsoft QuickC.[™]
- Fastest math, in-line 8087/80287 instructions, and floating-point calls.
- More complete support of proposed ANSI standard.
- Over 350 library functions, including a graphics library.

Microsoft CodeView

- Full OS/2 systems support. New.
- Debug applications of up to 128 MB under the OS/2 systems. New.
- Debug multithreaded programs and Dynamic Link Libraries. New.
- Source-level debugging for precise control over programs.
- Dynamic breakpoints in the source.
- Debug programs written in a variety of Microsoft languages. New.
- Full symbolic display of C structures. New.
- Interactively follow linked lists and nested structures. New.
- Watch variables, memory, registers, and flags.

Other Utilities

- Fast linking (twice as fast as the C 4.0 version linker).
- OS/2 incremental linker—up to 20 times faster than a full link. New.
- OS/2- and MS-DOS reconfigurable programmer's editor. New.

Everything about Microsoft® C Optimizing Compiler version 5.1 is dedicated to the professional programmer.

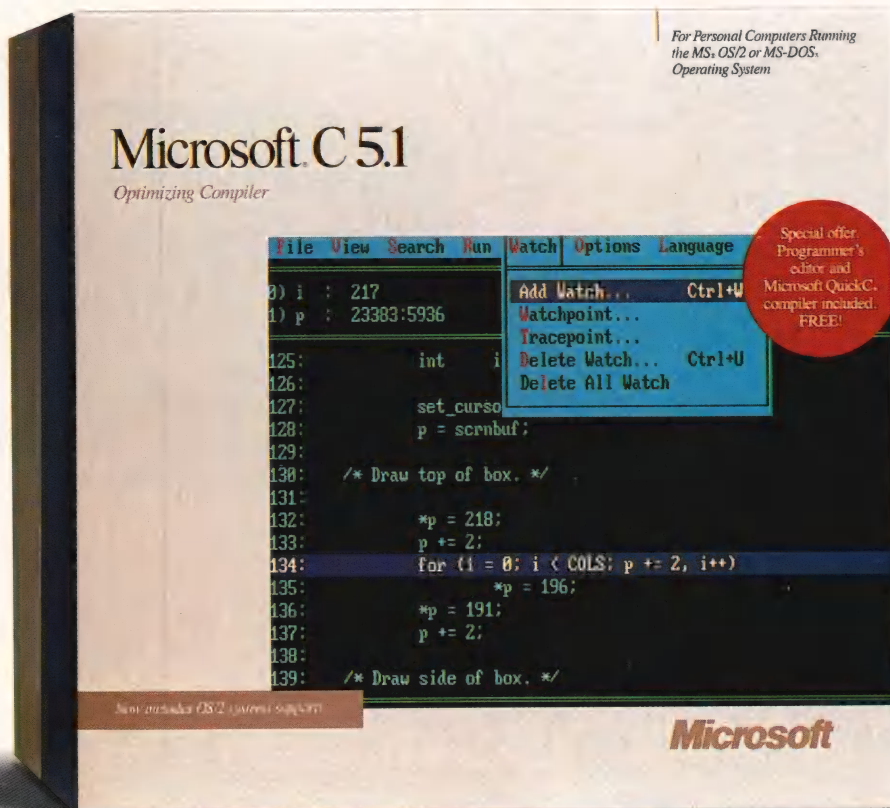
Fast code. Fast development. Fast debugging. And full support for both MS-DOS® and the OS/2 systems in a single package.

There's no faster C code on a PC, because powerful optimizations, such as in-line code generation and loop enregistering, generate executables that are compact and efficient. The documentation even teaches

you special coding techniques to squeeze every last bit of speed out of your code.

Fast code isn't all you get. Under MS® OS/2, the 640K barrier is gone so you can write C programs as large as a gigabyte. You can call the operating system directly. Create more responsive programs (multiple threads allow program operations to overlap). And build Dynamic Link Libraries (DLLs) that can be shared, saving valuable memory. DLLs also allow your main programs to be smaller, so they load faster.

Introducing the fastest possible way to create the fastest MS OS/2 programs possible.



Microsoft C Optimizing Compiler 5.1 Techbox

Compiler

- Optimizations that generate the fastest code for DOS and OS/2 systems.
 - In-line code generation.
 - Loop optimizations.
 - Elimination of common subexpressions.
- Full OS/2-system support to break the 640K barrier. New.
- Family API programs that run under DOS and the OS/2 systems. New.
- Write multithreaded programs and Dynamic Link Libraries. New.
- Small, medium, compact, large, and huge memory models.
- Mix models with NEAR, FAR, and HUGE keywords.
- Fast compilation (10,000 lines/minute) with Microsoft QuickC™.
- Fastest math, in-line 8087/80287 instructions, and floating-point calls.
- More complete support of proposed ANSI standard.
- Over 350 library functions, including a graphics library.

Microsoft CodeView

- Full OS/2 systems support. New.
- Debug applications of up to 128 MB under the OS/2 systems. New.
- Debug multithreaded programs and Dynamic Link Libraries. New.
- Source-level debugging for precise control over programs.
- Dynamic breakpoints in the source.
- Debug programs written in a variety of Microsoft languages. New.
- Full symbolic display of C structures. New.
- Interactively follow linked lists and nested structures. New.
- Watch variables, memory, registers, and flags.

Other Utilities

- Fast linking (twice as fast as the C 4.0 version linker).
- OS/2 incremental linker—up to 20 times faster than a full link. New.
- OS/2- and MS-DOS reconfigurable programmer's editor. New.

You can even write a single Family API program that runs under both MS-DOS and MS OS/2.

Microsoft Editor is the first reconfigurable text editor for programmers that lets you develop under MS-DOS and MS OS/2. Under MS OS/2, multitasking lets you edit one file while you compile another, which cuts development time. You can even generate multiple compiles that report errors directly back into your source code.

Microsoft CodeView® is the highly acclaimed

window-oriented source-level debugger that makes debugging fast and efficient. You can view program execution while you watch variables and register values change. And under MS OS/2 you can debug multithreaded applications, DLLs, and programs as large as 128 MB.

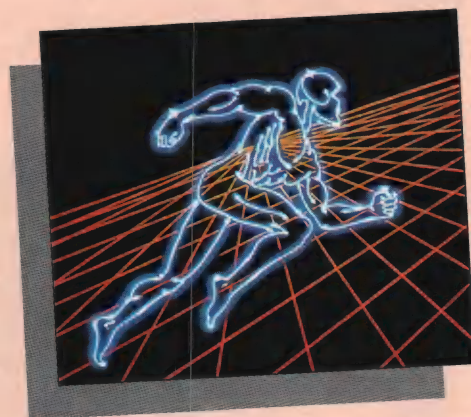
New Microsoft C Optimizing Compiler 5.1 for the professional programmer. It's all the speed you need. Call (800) 541-1261, **Microsoft** Department F34.

Introducing

FRONTRUNNER

*New...for dBASE III PLUS Users!
Fast...Resident...Powerful.
FrontRunner offers all this and more!*

- **CREATE MEMORY-RESIDENT dBASE III PLUS™ PROGRAMS** – FrontRunner™ is the first memory-resident applications development tool to contain a large subset of dBASE III PLUS commands and allows you to distribute RunTime™ applications.
- **dBASE III PLUS DATABASE AND INDEX FILE COMPATIBILITY** – Allows you to use FrontRunner immediately.
- **UNIQUE KEYBOARD FEATURE** – Bind commands or entire programs to a single Hotkey for rapid execution from within other applications.
- **PASTE COMMAND** – This powerful command allows you to extract data from your dBASE III PLUS files and paste it into your spreadsheet or word processing application.



Buy FrontRunner by June 30, 1988 and get a FrontRunner version of RunTime and an unlimited RunTime license for royalty-free applications. FrontRunner is not copy-protected and comes with a 30-day money-back guarantee.

The suggested retail price is \$195.

See your local Ashton-Tate dealer now. For more information, or the name of the dealer nearest you,

call (800) 437-4329, Ext. 556.*

*In Colorado, call (303) 799-4900, Ext. 556.



ASHTON-TATE

Trademarks / owner: dBASE III PLUS, RunTime, Ashton-Tate / Ashton-Tate Corporation; FrontRunner / Apex Software Corporation.
© 1988 Ashton-Tate Corporation. All rights reserved.

CIRCLE NO. 87 ON READER SERVICE CARD

algorithm should push the `prefix_1` for the `=` operator, which is the string `let`.

Finally, upon reaching the end of the input string, the stack elements are popped and the associated strings emitted and then the operand itself—`x`. At this point the output string is `let x`.

The scanning flow you have just observed is the basic pattern to Brown's algorithm. For each operand the algorithm scans ahead for operators that give rise to prefixes for the operand, pushing these prefixes onto the stack. When the forward scan is complete, the prefixes are popped from the stack and emitted and the scan proceeds to the next token.

The scan is now looking at the next operand—`y`. The same pattern is repeated. First, while the level is 0, the operator `!` is seen. This results in pushing `prefix_1` for `!`, which is a `-`. At the end of the string, the `=` is seen once again, and the `prefix_2` (`=`) is pushed on the stack. Now the forward scan terminates, the stack elements are popped and emitted, and finally the operand itself—`y`—is emitted. The output string is now `let x=-y`. Note that the use of the stack results in printing prefixes in the reverse of the order in which they are seen.

Once again the scan restarts, now at the operator `!`. When an operator is seen, the algorithm prints its suffix. In this case there isn't one, so you proceed. Next in line is the operand `z`. The usual routine is followed, resulting in the `prefix_2` for `/` being printed. The output string is now `let x=-y/z`. Because `z` is followed by the operator `/`, which has no suffix, you proceed. Finally, you encounter the operator `=`, which has the suffix `;`, which is emitted. The input string has been entirely scanned, so the algorithm is complete, with the output string being `let x=-y/z;`.

Simple as this example is, it shows that the algorithm is inefficient for large volumes of data. The tail of the string may be scanned once for each operand in the string. This inefficiency, however, is not evident in an application such as a spreadsheet, where expressions are limited in size, or in a line-by-line inter-

preter such as some forms of Basic.

Support Functions

The decompiler (see Listing One, page 82) uses several tools to provide supporting mechanics. These tools fall into three classes: input scanning, table handling, and stack handling.

Input scanning is done by the function `advance_to_next_elem()`. Scanning is much easier for a decompiler than it is for a compiler. Because it is scanning object code, not source, there is no need to han-

Bad things can happen, however, if an incorrectly formed expression is submitted for decompiling.

dle white space, ends of lines, numeric conversions, and so on. The format of the input stream is much more predictable. Even so, a real scanner can get much more complicated than that shown here because it will need to identify a variety of operand formats, including integers, floating point, cell references, and so on.

Accessing the transformation table is the next important set of functions. The table-handling functions are responsible for searching the table and returning pointers to associated strings. I have used a very simple if inefficient design based on linear searches. A slightly more sophisticated scheme might sort the table and use binary searches.

Finally, a stack is needed to store the results of the forward scans performed for each operator. These results are then unwound when the scan is complete. The stack functions shown can push and pop pointer-size objects. Although simple, this stack implementation is quite sufficient for this algorithm.

Usage

An application program sees only one externally visible function: `deparse(instr, outstr)`. The input string and output string cannot be the same. Some simple enhancements can be added if needed, such as a length check on the output string. Very little error checking should be needed in the decompiler. In theory at least, incorrectly formed expressions should have been caught by a compiler before they get to this stage. Bad things can happen, however, if an incorrectly formed expression is submitted for decompiling. In particular, if the RPN expression does not end with an operator, the algorithm will not terminate correctly, if it terminates at all.

In my implementation the transformation table is compiled into the program. Others, who desire more flexibility, could modify the code to pass a pointer to the table as one of the parameters, allowing use of multiple tables or making changes at run time.

Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063, or call 415-366-3600, ext. 221. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

Note

P.J. Brown, "More on the Re-creation of Source Code from Reverse Polish," *Software—Practice and Experience* 7 (1976): 545-551.

DDJ

(Listing begins on page 82.)

Vote for your favorite feature/article.
Circle Reader Service **No. 3.**

DR. DOBB'S TOOLBOOK OF 80286/ 80386 PROGRAMMING

by The Editors of Dr. Dobb's
Journal of Software Tools

The editors of *Dr. Dobb's Journal of Software Tools* have gathered the best 80286/80386 articles, updated and expanded them, and added new material to create this valuable resource for all 80X86 programmers.

Basic information has been compiled along with real world solutions. New and previously published articles on programming the 80386 microprocessor and its relatives, the 80387 math coprocessor, 82786 graphics coprocessor, and the 80286 16-bit processor are all included. You'll also find articles on moving old programs to the 32 bit 80386, reaping the benefits of the 386's memory-management abilities, creating and handling operating systems with multitasking and multiuser features, and more. All source code is available on disk.

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Galveston Dr., Redwood City, CA 94063 Or **CALL TOLL FREE 800-533-4372** (In CA 800-356-2002)

☐ **Yes ! Send me Dr. Dobb's Toolbook of 80286/80386 Programming.**

Book and Disk (MS-DOS) \$39.95 _____

Book only \$24.95 _____

CA residents add sales tax _____ %

Add \$2.95 per item for shipping _____

Total _____

☐ Check enclosed.

Make payable to M&T Books

Charge my ☐ Visa ☐ MC ☐ AmEx

Card No. _____ Exp. Date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

3052H

REAL-TIME PROGRAMS

Listing One (Text begins on page 18.)

```
/* Listing One -- Main routine for Solution A */

#include <stdio.h>                /* standard Unix I/O header */
#include <termio.h>              /* file/port control headers */
#include <fcntl.h>

#include <sys/types.h>           /* required headers for IPC */
#include <sys/ipc.h>
#include <sys/msg.h>

/* local constants */

#define TRUE (1==1)              /* boolean values */
#define FALSE (1==0)
#define SENSOR_1_DEV "/dev/tty45" /* tty port names for sensor ports */
#define SENSOR_2_DEV "/dev/tty46"
#define SENSOR_3_DEV "/dev/tty47"
#define TIMEOUT (10L)           /* timeout if no input received */
#define MAX_SENSOR_MSG (100)    /* maximum size of a sensor message */
#define MAX_CMD_MSG (100)       /* maximum size of a command message */

/* local function declarations */

void process_timeout();          /* process sensor read timeout */

/* module-wide data */

int timeout_id[3];              /* marktime timeout timer IDs */

/* Main routine for program */

main ()
{
    int
        cmd_size,               /* number of bytes in command message */
        msg_size,               /* number of bytes in sensor message */
        qid,                    /* message queue identifier */
        sensor_1_fd,            /* file descriptors for the sensors */
        sensor_2_fd,
        sensor_3_fd;

    char
        cmd_msg[MAX_CMD_MSG],   /* buffer for reading command queue */
        sensor_msg[MAX_SENSOR_MSG]; /* buffer for reading sensor data */

    /* open the sensor input files */

    sensor_1_fd=open(SENSOR_1_DEV,O_RDWR|O_NDELAY);
    sensor_2_fd=open(SENSOR_2_DEV,O_RDWR|O_NDELAY);
    sensor_3_fd=open(SENSOR_3_DEV,O_RDWR|O_NDELAY);

    /* open the command queue */

    qid=msgget(1,IPC_CREAT|0666);

    /* establish initial timeouts for each sensor */

    timeout_id[0]=marktime(TIMEOUT,(int*)NULL,process_timeout,(char*)1);
    timeout_id[1]=marktime(TIMEOUT,(int*)NULL,process_timeout,(char*)2);
    timeout_id[2]=marktime(TIMEOUT,(int*)NULL,process_timeout,(char*)3);

    /* loop forever */

    while (TRUE)
    {
        /* Read (or try) each sensor and process the input. */

        if ( (msg_size=read(sensor_1_fd, sensor_msg, MAX_SENSOR_MSG)) > 0)
            process_sensor (sensor_msg, msg_size, 1);
        if ( (msg_size=read(sensor_2_fd, sensor_msg, MAX_SENSOR_MSG)) > 0)
            process_sensor (sensor_msg, msg_size, 2);
        if ( (msg_size=read(sensor_3_fd, sensor_msg, MAX_SENSOR_MSG)) > 0)
            process_sensor (sensor_msg, msg_size, 3);

        /* check for input on the message queue */

        if ( (cmd_size=msgrcv(qid,cmd_msg,MAX_CMD_MSG,0,IPC_NOWAIT)) >= 0)
            process_cmd_msg (cmd_msg, cmd_size);

        /* delay the program before continuing loop */

        nap (3);
    }

    /* Function process_cmd_msg ()
    **
```



```

** This is a stub routine for handling command queue input.  It suspends
** timeouts while processing the input and then resumes them when it is
** done.
*/

```

```

static process_cmd_msg (msg, size)

```

```

char
    *msg;                /* INPUT: pointer to command message */

```

```

int size;                /* INPUT: size of *msg */

```

```

{
    timer_disable();
    /* (do some appropriate processing on the command) */
    timer_enable();
    return;
}

```

```

/* Function process_sensor()

```

```

**
** This is a stub routine for handling sensor input.  It cancels the
** outstanding timeout timer and reschedule a new timeout.  It also
** suspends timeout interrupts while it is processing the input.
*/

```

```

static process_sensor (msg, size, sensor_num)

```

```

char
    *msg;                /* INPUT: pointer to sensor data */

```

```

int
    size,                /* INPUT: size of *msg */
    sensor_num;          /* INPUT: sensor number */

```

```

{
    timer_disable();
    cancel_marktime (timeout_id[sensor_num-1]);
    /* (do some appropriate processing on the message) */
    timeout_id[sensor_num]=marktime(TIMEOUT,(int*)NULL,
        process_timeout, (char*)sensor_num);
    timer_enable();
    return;
}

```

```

/* Function process_timeout()

```

```

**
** This is a stub for the sensor input timeout timer.  It is called as a
** completion routine from marktime(), which passes the sensor number
** as an argument.  The function suspends timeouts while it processing
** the error and resumes processing when it is done.  It also resets the
** timeout before exiting, something that a real program may or may not
** want to do in real life, depending on the application.
*/

```

```

static void process_timeout (sensor_num)

```

```

char
    *sensor_num;          /* INPUT: sensor number which has */
                        /* timed out (declared char* because */
                        /* that's what marktime() calls */
                        /* it with.  Value is really int */

```

```

{
    timer_disable();
    /* (do some appropriate processing on the error) */
    timeout_id[(int)sensor_num]=marktime(TIMEOUT,(int*)NULL,
        process_timeout, sensor_num);
    timer_enable();
    return;
}

```

End Listing One

Listing Two

```

/* Listing Two -- nap() */

#include <stdio.h>        /* standard Unix I/O header */

#include <termio.h>       /* port file control headers */
#include <fcntl.h>

#define TRUE (1==1)      /* boolean constants */
#define FALSE (1==0)

/* Function nap()
**
** This function provides a program delay function with resolution
** to a tenth of a second.  It works by opening a file to /dev/clk
** (which should be linked to some unused /dev/tty), setting the
** input parameters to non-canonical, and setting the VTIME value
** to the passed argument.  It then does a read on the file which
** will time out after the indicated delay time.

```

(continued on next page)

Q. How many programmers does it take to maintain a MAKE dependency file?

A. NONE! If you use VersiMAKE™

VersiMAKE™ is a full-featured MAKE utility that includes:

- **Dependency Generation**
Derives your system's dependencies, through analysis of its C and MASM source files. Say goodbye to manual maintenance of MAKE dependency files!
- **Wild Card File Name Matching**
Analyzes an entire collection of source files with a single statement.
- **Nested Include Files**
Handles standard C and MASM "include" conventions, and the INCLUDE environment variable.
- **Powerful Macro Facilities**
Built-in macros, user-defined macros, and environment variables.
- **Analytical Reports**
Shows the entire Include file hierarchy for each source file analyzed, and all of the parent source files for each Include file.

Q. How many programmers does it take to trace a symbol thru your system?

A. ONE! If you use VersiCREf™

VersiCREf™ is a unique utility that creates a Master Cross-Reference of your entire system.

- **Multi-Lingual**
Handles C, assembler, or both.
- **Flexible**
File names with line numbers, or file names alone. Global and local symbols, or globals alone.
- **Powerful**
Easily handles systems containing 100 source files or more.



VersiMAKE™ \$125
VersiCREf™ \$75
Both \$150
(Outside U.S., add \$5 for shipping and handling)

800-334-4096
(In NJ, 609-871-0202)
MC/VISA/AMEX accepted

SUMMIT INFORMATION SYSTEMS, INC.

73 East Lane, Willingboro, NJ 08046

CIRCLE NO. 233 ON READER SERVICE CARD

The Experts' Choice



"The best disk optimizer I've seen, it quickly unfragments your hard disk and keeps it as fast as it is supposed to be."

Bernie Zilbergeld
Computer Currents

"Vopt is fast, safe, effective, and even fun to use. What more could you want?"

Glenn Hart
PC Magazine

"There are several disk management programs available, but the one I use is Golden Bow's Vopt."

Jerry Pournelle
Byte

"In three years of rating software, I've never given a product a 10—until now."

Vincent Flanders
Access 88

"Vopt is very, very FAST. Golden Bow Systems has a winner here. No choice. In stand-alone disk optimizers, Vopt is Vbest."

John G. Scherb
Tokyo PC Newsletter

Vopt is the fast, safe, disk organizer that unfragments your disk files to improve the performance of your hard and floppy disks.

Vopt is loaded with additional programs that test and report on the efficiency of your system.

Call toll free and receive a free demo disk that will show just how fast Vopt will work for you!

\$59.95 \$3 shipping/handling
CA add 6% sales tax

GOLDEN BOW SYSTEMS



2870 Fifth Avenue
Suite 201
San Diego, CA 92103
800/284-3269

Vopt is a trademark of Golden Bow Systems.

CIRCLE NO. 132 ON READER SERVICE CARD

REAL-TIME PROGRAMS

Listing Two (Listing continued, text begins on page 18.)

```

*/
void nap (delay)

int
    delay;                /* INPUT: delay in tenths of a second */
{
    static int
        clock_opened=FALSE;    /* has the clock device been opened? */

    static int
        fd;                    /* clock file descriptor */

    int
        flags;                /* file control flags */

    char
        buff[10];            /* dummy read buffer */

    struct termio
        clock_termio;        /* terminal I/O parameters */

    /*
    ** the first time through the routine, open the clock port
    ** and set the port parameters.
    */

    if (!clock_opened)
    {
        fd=open("/dev/clk",O_RDONLY|O_NDELAY);
        ioctl (fd, TCGETA, &clock_termio);
        clock_termio.c_cflag |= CLOCAL;
        clock_termio.c_lflag &= ICANON;
        ioctl (fd, TCSETA, &clock_termio);
        flags = fcntl (fd, F_GETFL, 0) & O_NDELAY;
        fcntl (fd, F_SETFL, flags);
        clock_opened = TRUE;
    }

    /* set the VTIME delay to the indicated value */

    ioctl (fd, TCGETA, &clock_termio);
    clock_termio.c_cc[VMIN] = 0;
    clock_termio.c_cc[VTIME] = delay;
    ioctl (fd, TCSETAF, &clock_termio);

    /* perform the dummy read */

    read (fd, buff, 10);

    return;
}
    
```

End Listing Two

Listing Three

```

/* Listing Three -- marktime() and related functions */

#include <stdio.h>                /* standard input/output include */
#include <sys/signal.h>          /* signal definitions */

#define MAX_ELEMENT 10          /* maximum number of queued events */
#define TRUE (1==1)
#define FALSE (1==0)

struct tmq                      /* timer queue element structure */
{
    unsigned long time;          /* expiration time (UNIX) of element */
    void (*ast)();              /* user ast to call at expiration */
    char *arg;                  /* value passed to ast() (if called) */
    struct tmq *next;           /* pointer to next element */
    int *flag;                  /* event count to bump at expiration */
};

static struct tmq
{
    *queue_free,                /* first available element in queue */
    *queue_head,               /* first element in clock queue */
    *queue_tail,               /* last element in clock queue */
    timer_queue[MAX_ELEMENT];  /* table of queue elements */
};

static int
{
    q_busy = FALSE,            /* queue update in progress */
    q_enabled = FALSE,         /* queue countdown operations enabled */
    queue_init = FALSE;        /* queue initialized flag */
};

static unsigned long
    next_event;                /* last known value of expiration */
/*
    time of head element (may
    change during ast) */

extern unsigned long
    time();                    /* get Unix time */
    
```

(continued on page 60)

C the Improvement.

NOW
VERSION 3.3
For MS-DOS and OS/2!



Now the Lattice C Compiler takes you where it's never gone before. With Version 3.3, it works on *two* operating systems: MS-DOS *and* OS/2!

You may now use Version 3.3 on an MS-DOS system to create programs that run under OS/2 protected mode. Or vice versa. A simple "switch" has been put into the compiler to let you generate code for either system or both.

New improved standards...

Version 3.3 is fully compliant with the latest ANSI C standards. It also has improved embedded system support, enhancements to the standard libraries and a host of other compiler advances too numerous to compile.

At a new improved price and value!

The suggested retail for Lattice C Version 3.3 is only \$450. And 3.3 also includes "family" versions of the Lattice Screen Editor (LSE) and the Lattice C-SPRITE™ symbolic debugger, compatible with both MS-DOS and OS/2 systems, at no charge.

C for yourself why Lattice is the professional programmer's choice for serious MS-DOS and OS/2 programming.



Lattice, Incorporated
2500 S. Highland Avenue
Lombard, IL 60148
Phone: 800/533-3577
In Illinois: 312/916-1600

Lattice is a registered trademark of Lattice, Incorporated. MS-DOS is a registered trademark of Microsoft Corp. OS/2 is a registered trademark of International Business Machines Corp.

CIRCLE NO. 142 ON READER SERVICE CARD

REAL-TIME PROGRAMS

Listing Three (Listing continued, text begins on page 18.)

```
extern unsigned int
alarm();                /* set system alarm clock */

void
queue_ast();            /* internal asynchronous trap */

/* Function marktime()
**
** This function inserts an element into the timer queue (performing
** queue initialization if necessary). If the new element is at the
** top of the queue, it will reset the alarm clock.
**
** Return values:
**      -1      queue full
**      >=0     element ID
**/

int marktime (time_of_event, flag, user_ast, user_arg)

unsigned long
time_of_event;          /* INPUT: seconds until event */

char
*user_arg;              /* INPUT: user-supplied argument (in */
                        /* reality, this could be a */
                        /* pointer to any data type, but */
                        /* char* is a good enough */
                        /* description */

void
(*user_ast)();          /* INPUT: user function to call at */
                        /* expiration (may be NULL) */

int
*flag;                  /* INPUT: pointer to flag to make */
                        /* TRUE on expiration (may be NULL) */

{
int
element,                /* offset into timer_queue */
found_slot,             /* loop termination flag */
ret_val;                /* >=0 element ID, -1 queue full */

register struct tmq
*new_element,           /* pointer to newly added element */
*last_ptr,              /* previous pointer into timer_queue */
*q_ptr;                 /* pointer into timer_queue */

/* if the queue is uninitialized, then initialize it */

if (!queue_init)
{
queue_free = &timer_queue[0];
for (element=0; element<MAX_ELEMENT; element++)
timer_queue[element].next = &timer_queue[element+1];
timer_queue[MAX_ELEMENT-1].next = NULL;
queue_head = NULL;
queue_tail = NULL;
q_busy = FALSE;
q_enabled = TRUE;
signal (SIGALRM, queue_ast);
queue_init = TRUE;
}

/* insert the new element into the linked list */

if ( (new_element=queue_free) != NULL)
{
q_busy = TRUE;
queue_free = queue_free->next;
new_element->time = time_of_event + time((long*)0);
if (flag != NULL)
{
new_element->flag = flag;
*flag = FALSE;
}
new_element->ast = user_ast;
new_element->arg = user_arg;
q_ptr = queue_head;
last_ptr = queue_head;
found_slot = FALSE;
while ( (q_ptr!=NULL) && !found_slot)
{
if (new_element->time < q_ptr->time)
found_slot = TRUE;
else
{
last_ptr = q_ptr;
q_ptr = q_ptr->next;
}
}
}
```



```

    }
    /* if the new element is first, then reset alarm for this element */
    if (q_ptr == queue_head)
    {
        new_element->next = queue_head;
        queue_head = new_element;
        next_event = new_element->time;
        if (q_enabled)
            alarm ((unsigned int)(new_element->time - time((long*)0)));
    }
    else
    {
        new_element->next = q_ptr;
        last_ptr->next = new_element;
    }
    ret_val = new_element - timer_queue;
    q_busy = FALSE;
}
else
    ret_val = (-1);

return (ret_val);
}

/* Function queue_ast()
**
** This function is called when the clock reaches the time
** at the head of the timer queue. It sets the indicated
** flag (if non-NULL), and removes the head element
** from the queue. It reschedules the internal timeout to the
** time of the new queue head element, if one exists. (If the
** time of the next element is less than the present time, the
** function schedule the event in one second.) Finally,
** if the user-supplied ast is non-NULL, it calls that routine,
** passing the user-supplied argument.
**
** The function checks the q_busy flag (set by marktime() and
** cancel_marktime()) to verify that the program isn't in the
** middle of a queue update. If the flag is TRUE, queued_ast()
** reschedules itself 1 second from now, rather than attempting
** to hack at the queue blindly.
**
*/
static void queue_ast ()
{
    register struct tmq
        *q_ptr; /* temporary queue pointer */

    unsigned int
        nexttime; /* seconds until next timeout */

    /*
    ** If it is safe to fiddle with the queue, pull out the next
    ** element and schedule the next timeout, if any.
    */
    if (!q_busy)
    {
        q_ptr = queue_head;
        queue_head = q_ptr->next;
        q_ptr->next = queue_free;
        queue_free = q_ptr;
        if (queue_head != NULL)
        {
            if (q_enabled)
            {
                if ( (nexttime=(unsigned int)queue_head->time -
                    time((long*)0)) > 0)
                    alarm (nexttime);
                else
                    alarm (1);
            }
        }
        /* set the user's flag, if any is specified */
        if (q_ptr->flag != NULL)
            *q_ptr->flag = TRUE;

        /* call the user's completion routine, if any specified */
        if (q_ptr->ast != NULL)
            (*q_ptr->ast)(q_ptr->arg);
    }

    /* if the queue is busy, reschedule the timeout for later */
    else
    {
        if (q_enabled)

```

(continued on page 63)

Introducing

NANODISK

"Disk Cache for the IBM PC"

Make your floppy drive and hard disk run close to RAM disk speeds. Dramatic speed improvement for most programs. Supports cache of any size in main or expanded memory.

Requires IBM PC/XT/AT or true clone.

only **\$29.95**

MultiDos Plus

"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 DOS programs concurrently.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
 - * Intertask message communication.
 - * Task control by means of semaphores.
 - * 256 priority levels.
 - * Suspend task for specified interval.
 - * Spawn and terminate external and internal tasks.
 - * Disable/enable multitasking.
 - * and more!

Requires IBM PC/XT/AT or true clone, and enough memory to hold **MultiDos Plus** (48 KB) and all your application programs.

\$24.95 or **\$99.95**

With source code
(Written in Lattice C
and Microsoft Assembler.)

Outside USA add \$5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350, or send check or money order (Drawn on U.S. Bank Only) to:

NANOSOFT

13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

CIRCLE NO. 170 ON READER SERVICE CARD

New Prices

OS/2

WINDOWS FOR DATA[®]

MULTI-LEVEL
MENU SYSTEM

NESTED
POP-UP FORMS

SCROLLABLE
REGION

CHOICE LIST

Invoices: Create Review Print Exit

INVOICE

Invoice No.: 008784 Date: 12/03/87 Time: 16:43:15

Customer: William Jones
Innovative Software
351 Bulletin Avenue
Needham, MA 02194
(617) 394-5512

Search for customer record? (Y/N): N
Enter customer information? (Y/N): N
Enter billing address? (Y/N): N
Enter marketing information? (Y/N): N

No.	PRODUCT	DESCRIPTION	QUANTITY	PRICE	AMOUNT
5	WDMS	Windows for Data Microsoft	10	295.00	2950.00
6	WDLA	Windows for Data Lattice	5	295.00	1475.00
7	WDTC	Windows for Data Turbo C	5	295.00	1475.00
8	WDXE	Windows for Data XENIX	2	795.00	1590.00
9			0	0.00	0.00

Subtotal: 11325.00
Shipping: 0.00
TOTAL: 11325.00
Payment: 0.00

Cursor keys scroll, ENTER selects and ESC exits choice menu

CLOCK

POP-UP
WINDOW

RUNNING
TOTALS

MESSAGE
WINDOW

If you program in C, take a few moments to learn how Windows for Data can help you build a state-of-the-art user interface.

- ✓ Create and manage menus, data-entry forms, context-sensitive help, and text displays — all within windows.
- ✓ Develop window-based OS/2 programs right now, without the headaches of learning OS/2 screen management. Run the same source code in PC DOS and OS/2 protected mode.
- ✓ Build a better front end for any DBMS that has a C-language interface (most popular ones do).



FROM END TO BEGINNING

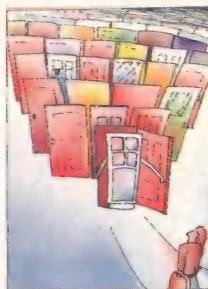
Windows for Data begins where other screen packages end, with special features like nested pop-up forms and menus, field entry from lists of choices, scrollable regions for the entry of variable numbers of line items, and an exclusive built-in debugging system.

NO WALLS

If you've been frustrated by the limitations of other screen utilities, don't be discouraged. You won't run into walls with Windows for Data. Our customers repeatedly tell us how they've used our system in ways we never imagined — but which we anticipated by designing Windows for Data for unprecedented adaptability. You will be amazed at what you can do with Windows for Data.

YOU ARE ALWAYS IN CHARGE

Control functions that you write and attach to fields and/or keys can read, compare, validate, and change the data values in all fields of the form. Upon entry or exit from any field, control functions can call up subsidiary forms and menus, change the active field, exit or abort the form, perform almost any task you can imagine.



OUR WINDOWS WILL OPEN DOORS

Our windows will open doors to new markets for your software. High-performance, source-code-compatible versions of Windows for Data are now available for PC DOS, OS/2, XENIX, UNIX, and VMS. PC DOS

versions are fully compatible with Microsoft Windows. **No royalties.**

MONEY BACK GUARANTEE

You owe it to yourself and your programs to try Windows for Data. If not satisfied, you can return it for a full refund.

Prices: PC DOS \$295, Source \$295. OS/2 \$495. XENIX \$795. UNIX, VMS, please call.

Call: (802) 848-7731

Telex: 510-601-4160 VC SOFT

ext. 31

FAX 802-848-3502



**Vermont
Creative
Software**

21 Elm Ave.
Richford,
VT 05476

REAL-TIME PROGRAMS

Listing Three (Listing continued, text begins on page 18.)

```

    alarm (1);
}

/* reset signal catcher */

signal (SIGALRM, queue_ast);

return;

}

/* Function cancel_marktime()
**
** This function removes the specified timer request from the
** timer queue.
**
** Return values:
**     >=0      time remaining till expiration
**     <0xFFFFF no such element
**
** If the queue hasn't been initialized, the function returns
** 'no such element'. It does NOT initialize the queue.
**
** The element ID, used for identifying the event to be canceled
** is returned by marktime.
**
*/

unsigned long cancel_marktime (id)

int
    id;                /* INPUT: element id (returned from */
                      /* marktime) */

{
    register struct tmq
        *last_ptr,      /* previous pointer into timer_queue */
        *q_ptr;         /* pointer into timer_queue */

    int
        found;          /* loop termination flag */

    unsigned long
        ret_val;         /* >=0 for success, <0 for failure */

    unsigned int
        neyvertime;      /* time until next event expiration */

    /* make sure that the queue is initialized */
    if (queue_init)
    {
        q_busy = TRUE;
        /* make sure that the ID is legitimate */
        if ( (id >= 0) && (id < MAX_ELEMENT) )
        {
            /*
            ** Traverse the event queue until we find the requested element.
            ** This ensures that the element has really been used and also
            ** gives us the pointers for relinking the list.
            */
            for (q_ptr=queue_head, last_ptr=q_ptr, found=FALSE;
                q_ptr!=NULL && !found;
                last_ptr=q_ptr, q_ptr=q_ptr->next)
            /* do we have a match? */
            if (q_ptr == &timer_queue[id])
            {
                ret_val = q_ptr->time - time((long*)0);
                /*
                ** if the cancelled element was at the head, then
                ** reschedule the next timeout, if any exist.
                */
                if (q_ptr==queue_head)
                {
                    queue_head = q_ptr->next;
                    if (queue_head != NULL)
                    {
                        if (q_enabled)
                        {
                            if ((nexttime=(unsigned int)queue_head->time
                                - time((long*)0)) > 0)
                                alarm (nexttime);
                            else
                                alarm (1);
                        }
                    }
                }
                else
                    alarm (0);
            }
            else
                last_ptr->next = q_ptr->next;
            q_ptr->next = queue_free;
            queue_free = q_ptr;
            found = TRUE;
        }
    }
}

```

(continued on page 65)

PC/Forms Screen Management Software **SLASHES** Development Time!

- PC/Forms takes the hassle out of screen design, screen management and input data validation.
- Forms are created & maintained using a form editor, loaded and processed at run time via the PC/Forms run time library.
- This is not a code generator.
- There is no memory resident form manager.
- Forms can be from one to ten screens in length.
- Form dimensions are adjustable (for windowing).

Form Editor Features

- Full control over foreground & background video attributes.
- Access to the extended (graphics) char. set.
- Line and box drawing.
- Define and modify field attributes:
 - Field Name
 - Field Order
 - Edit Mask
 - Default
 - Auto Tab
 - Must Respond
 - Numeric Test
 - Right Justify
 - Echo Data
 - Display Only
 - Upper Case
 - Warning Only
 - Test Range
 - Data Type
 - Numeric Precision
- Test form utility.
- Generate program shell utility.
- Field reorder utility.
- Temporary exit to DOS.
- Compile form definitions to .OBJ files.

Run Time Library

- Routines are color (CGA, EGA, VGA) / monochrome independent.
- Forms are processed in dynamic memory.
- User written validation routines can be linked to fields.
- String, Byte, Integer, Long, Real, and Double data types are supported.
- Scrolling fields.
- Run time library source code included.
- Run time library includes (plus others):
 - load_form()
 - put_field()
 - get_form()
 - release_form()
 - put_form()
 - clear_form_buffer()
 - display_form()
 - get_field()
 - alter_field_attr()
- No royalties.

System Requirements

- IBM PC/XT/AT/PS2 or compatible.
- PC-DOS or MS-DOS 2.0 or later

Ordering Information

MC/VISA/Checks.

Demo disk available.

Call for shipping

dates on other ver-

sions.

Prices

Turbo Pascal	\$99.95
Microsoft Pascal . .	\$149.95
Microsoft C	\$149.95
Lattice C	\$149.95
Turbo C	\$149.95



1-800-338-6754
(US)
1-216-292-0224
(OH)

P.O. Box 22216 • 23500 Mercantile Rd.
Beachwood, OH 44122

Hours: Mon-Fri: 7:30 a.m. - 4:30 p.m., EST.

CIRCLE NO. 134 ON READER SERVICE CARD

PAINLESS WINDOWS.

Windows. Data Entry. Menus.
Finally, a C programmers' tool that makes
them as easy to use as *printf()*.
With Greenleaf DataWindowsTM,
you move in quantum leaps!

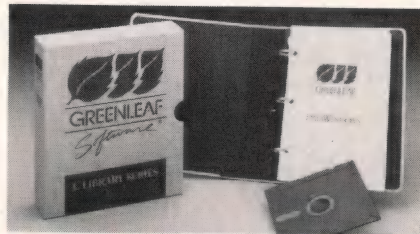
Snazzy Window Treatments

DataWindows represents an important breakthrough in C programming tools. It sets you free so you can create exciting programs quickly and easily, saving both time and money! Developed to work with the IBM PC, XT, AT, compatibles, and MSDOS or PCDOS, DataWindows is a carefully toolled system of C functions which will jazz up your programs with unprecedented efficiency.

Greenleaf DataWindows is integrated windows, transaction data entry, pop-up, pull-down, and Lotus style menu systems with:

- **Screen Management.** You don't have to remember what's on the display or the sequence in which you put it there. DataWindows does the grunt work. There are no restrictions.
- **Transaction Data Entry.** Data entry windows can have any number of fields with sophisticated options for reading many data types. Calls are made to help, validation, and other functions. Full featured text editing, protected and mandatory fields, dBASE type picture strings, context sensitive help, validation of fields and transactions, redefinable keys, password entry, attribute control, keyboard idle and much more.
- **Device Independence.** It detects the type of display adapter your computer is using and adjusts to it automatically for CGA, EGA, or monochrome. Logical video attributes are easy to use for color or monochrome.
- **Compatibility.** Runs with Microsoft Windows and IBM TopView.
- **The Greenleaf Tradition of Quality.** Reliable products. Professional documentation that gets you up and running quickly and keeps you there. Reference card. Newsletter and Bulletin board.

IBM, Microsoft & dBase, are registered trademarks of International Business Machines, Microsoft Corporation & Ashton-Tate respectively. PCDOS, IBM PC, XT, AT, & TopView are trademarks of IBM; MSDOS and Microsoft Windows are trademarks of Microsoft Corporation.



Stop Window Shopping

Order Today. Or call toll free for a free demo of the windows library that makes all the others obsolete.

Order any of these high performance tools by calling your dealer or 1-800-523-9830 today. Specify compiler when ordering. Add \$8 for UPS second day air, or \$5 for ground. Texas residents add sales tax. MasterCard, VISA, P.O., check, COD. In stock, shipped next day.

Greenleaf DataWindows	\$225
DataWindows Source Module	\$225
The Greenleaf Comm Library v2.0	\$185
The Greenleaf Functions v3.0	\$185
Digiboard Comm/4-II	\$325
Digiboard Comm/8-II	\$535



16479 Dallas Parkway, Suite 570
Dallas, TX 75243

Call Toll Free
1-800-523-9830
In Texas and Alaska, call
214-248-2561

Window Dressings

- **Simple or Complex Windows.** Up to 254 powerful overlaid windows simultaneously, all with just one kind of window to remember! Yet any window can be from one character to 32K!
- **Easy Window Operations.** DataWindows lets you move, zoom, frame, title, change colors, titles, frames, size, location, and make windows visible or invisible at will! Functions set cursor, attributes, and write data to any window or "current window". Word wrap, auto scroll, keyboard functions.
- **Write to Any Window Any Time.** Windows may be visible, overlaid, or invisible, and you can write to them anyway. What you write will be seen when the windows become visible.
- **DataWindows is fast!** It writes directly to video memory (in some modes).
- **Easy to save!** Any window, complete with attributes, can be saved on disk quickly and efficiently.
- **Source code available. No royalties.**

Also from Greenleaf:

The Greenleaf Functions v3.0

The most complete, mature C language function library for the IBM PC, XT, AT and close compatibles. Includes over 225 functions — DOS, disk, video, color text and graphics, string, time/date, keyboard, disk status and Ctrl-Break functions plus many more.

The Greenleaf Comm Library

Our 2.0 version is the hottest communications facility of its kind. Over 120 functions — ring buffered, interrupt driven asynchronous communications for up to 16 ports simultaneously with XMODEM, XON/XOFF, many many sophisticated features.

We support all popular C compilers for MSDOS/PCDOS: Microsoft, Lattice, Computer Innovations, Aztec, DeSmet, and others.

REAL-TIME PROGRAMS

Listing Three (Listing continued, text begins on page 18.)

```

    }
    /* did we ever find a match? */
    if (!found)
        ret_val = (-1);
    else
        ret_val = (-1);
    q_busy = FALSE;
}
} else
    ret_val = (-1);

return (ret_val);

}
/* Function timer_enable()
**
** This function enables normal timer queue operations. If
** there is an element on the top of the timer queue, it sets
** up the appropriate alarm; otherwise, just marks the queue
** as enabled. If the time of the next event has already passed,
** it will schedule it to happen in one second (prevents unexpected
** interrupt).
**
*/

void timer_enable()
{
    unsigned int
        nexttime;                /* seconds till next timeout */

    q_enabled = TRUE;
    if (queue_head != NULL)
    {
        if ( (nexttime=(unsigned int)queue_head->time - time((long*)0)) > 0 )
            alarm (nexttime);
        else
            alarm (1);
    }
    else
        alarm (0);

    return;
}

/* Function timer_disable()
**
** This function disables normal timer queue operations. If
** there is an element on the top of the timer queue, it cancels
** the alarm() timer; otherwise, just marks the queue as disabled.
**
*/
void timer_disable()
{
    q_enabled = FALSE;
    if (queue_head != NULL)
        alarm (0);

    return;
}

```

End Listing Three

Listing Four

```

/*Listing Four -- Sensor message definition */

/* sensor message command queue message structure */

struct message_rec
{
    long         mtype;          /* message type */
    int          func;           /* message function code */
    int          sensor_num;     /* sensor number */
    char         data[100];      /* message data */
};

/* sensor command queue function codes */

#define SENSOR_INPUT (1)
#define SENSOR_TIMEOUT (2)
#define SENSOR_COMMAND (3)

```

End Listing Four

Listing Five

```

/* Listing Five -- Main routine for Solution B */

#include <stdio.h>                /* standard Unix I/O header */

#include <termio.h>               /* file/port control headers */
#include <fcntl.h>

#include <sys/types.h>            /* required headers for IPC */
#include <sys/ipc.h>

```

(continued on next page)

MetaWINDOW

Power Graphics for your PC!

PC TECH JOURNAL

"Product of the Month"

"... a technological tour de
force for fast PC graphics."

NO ROYALTIES!

MetaWINDOW is a
Power Graphics

NEW! - QuickWINDOW/C
All the features of MetaWINDOW
for Microsoft Quick/C! - \$95*

Unparalleled Performance!

MetaWINDOW provides an expanded
set of graphic drawing functions,
plus the added functionality and
performance required for designing
multi-window desktop applications.

- auto-cursor tracking
- pull-down menus
- pop-up windows
- comprehensive
graphic functions
- multiple fonts



10 Point 12 Point
Bold Italic

Enhanced Features!

- Display multiple bitmap or
"filled-outline" fonts.
- Face fonts for bold, italic, under-
line or strike-out stylings.
- Full "RasterOp" transfer
functions for writing, erasing,
rubberbanding or dragging:
lines, text, icons, bit images
and complex objects.
- Create pop-up menus,
windows and icons.
- Supports IBM's new PS/2 VGA
and MCGA graphics.

MetaWINDOW comes complete with
language bindings for 20 popular C,
Pascal and Fortran compilers, plus
dynamic runtime support for over 50
graphics adaptors and input devices.

MetaWINDOW

Advanced Graphics Toolkit

4 disks, 3 260 page manuals - \$195*

NEW! - TurboWINDOW/Pascal

All the features of MetaWINDOW for
Borland Turbo Pascal Ver. 4! - \$95*
* Plus \$5.00 shipping and handling

TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550



METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road
Scotts Valley, CA 95066

CIRCLE NO. 156 ON READER SERVICE CARD

NROFF/PC™

The REAL Thing for DOS

NROFF/PC is a complete text formatting system for MS-DOS systems. Including:

NROFF The *powerful* UNIX text formatter

TBL A tool to assist with the layout of tabular material in *Nroff* documents

MM A comprehensive *Nroff* macro package for preparing books and technical manuals

NEQN A tool for describing mathematical equations in *Nroff* documents

- All tools are a complete port from the AT&T Documentor's Workbench 2.0
- It's **Fast!** We've modified *Nroff* especially for DOS for lightning speed
- Supports any Dot Matrix printer and many laser printers
- Specially Priced At **\$99**
- A complete *Troff* typesetting system is available **NOW** for LaserJet and PostScript printers on MS-DOS for \$695, XENIX and Microport UNIX for \$795.



Elan Computer Group, Inc.
410 Cambridge Ave., Suite A
Palo Alto, CA 94306
(415) 322-2450

Visa and MasterCard Accepted

Unix is a trademark of AT&T

MS-DOS and Xenix are trademarks of Microsoft

REAL-TIME PROGRAMS

Listing Five (Listing continued, text begins on page 18.)

```
#include <sys/msg.h>

#include "sensor.h"          /* defines sensor messages */

/* local constants */

#define TRUE (1==1)          /* boolean values */
#define FALSE (1==0)

/* Main routine for program */

main ()
{
    int
        cmd_size,            /* number of bytes in command message */
        qid;                 /* message queue identifier */

    struct message_rec
        cmd_msg;              /* buffer for reading queue message */

    /* open the command queue */

    qid=msgget(1,IPC_CREAT|0666);

    /* loop forever */

    while (TRUE)
    {
        /* wait for a new message on the command queue */

        /* check for input on the message queue */

        cmd_size = msgrcv (qid, &cmd_msg, sizeof(cmd_msg), 0, 0);
        switch (cmd_msg.func)
        {
            case SENSOR_INPUT :
                process_sensor (cmd_msg.data, cmd_size, cmd_msg.sensor_num);
                break;
            case SENSOR_TIMEOUT :
                process_timeout (cmd_msg.sensor_num);
                break;
            case SENSOR_COMMAND :
                process_cmd_msg (&cmd_msg, cmd_size);
                break;
        }
    }

    /* Function process_cmd_msg ()
    **
    ** This is a stub routine for handling command queue input.
    */

    static process_cmd_msg (msg, size)

    char
        *msg;                 /* INPUT: pointer to command message */

    int
        size;                 /* INPUT: size of *msg */

    {
        /* (do some appropriate processing on the command) */
    }

    /* Function process_sensor
    **
    ** This is a stub routine for handling sensor input.
    */

    static process_sensor (msg, size, sensor_num)

    char
        *msg;                 /* INPUT: pointer to sensor data */

    int
        size,                 /* INPUT: size of *msg */
        sensor_num;           /* INPUT: sensor number */

    {
        /* (do some appropriate processing on the message) */
    }

    /* Function process_timeout()
    **
    ** This is a stub for the sensor input timeout timer.
    */
}
```



```
static process_timeout (sensor_num)

int
    sensor_num;          /* INPUT: sensor which has timed out */
{
    /* (do some appropriate processing on the error) */
}
```

End Listing Five

Listing Six

```
/* Listing Six -- Sensor input process (one per sensor) */

#include <stdio.h>          /* standard Unix I/O header */
#include <termio.h>         /* file/port control headers */
#include <fcntl.h>
#include <sys/types.h>      /* required headers for IPC */
#include <sys/ipc.h>
#include <sys/msg.h>        /* sensor command message structure */
#include "sensor.h"

#define TRUE (1==1)        /* boolean true and false */
#define FALSE (1==0)

#define TIMEOUT (10L)      /* sensor timeout delay */

main(argc,argv)
int
    argc;                  /* INPUT: number of command line arguments */
char
    *argv[];               /* INPUT: pointers to command line arguments */
{
    int
        flags,             /* file control flags */
        marktime_id,       /* timeout timer ID */
        qid,               /* IPC message queue ID */
        sensor_fd,         /* file descriptor for sensor port */
        timeout;           /* marktime timeout flag */

    char
        filename[20];       /* name of port file */

    struct message_rec
        sensor_msg;         /* buffer for sensor message */

    struct termio
        clock_termio;       /* terminal I/O parameters */

    /* open the sensor port */

    sprintf (filename, "/dev/tty%d", atoi(argv[1]));
    sensor_fd=open(filename,O_RDWR);

    /* open IPC queue */

    qid=msgget(1,IPC_CREAT|0666);
    sensor_msg.mtype = 1;
    sensor_msg.sensor_num = atoi(argv[1]);

    /* main loop */

    while (TRUE)
    {
        /* set up timeout */

        marktime_id=marktime(TIMEOUT,&timeout,(void*)NULL,(char*)NULL);

        /* read until data received or timeout timer expires */

        read (sensor_fd, sensor_msg.data, sizeof(sensor_msg.data));
        if (!timeout)
        {
            /* got data -- cancel timeout and pass to main program */
            cancel_marktime (marktime_id);
            sensor_msg.func = SENSOR_INPUT;
            msgsnd (qid, &sensor_msg, sizeof(sensor_msg), 0);
        }
        else
        {
            /* timeout -- inform main program */
            sensor_msg.func = SENSOR_TIMEOUT;
            msgsnd (qid, &sensor_msg, sizeof(sensor_msg), 0);
            marktime_id=marktime(TIMEOUT,&timeout,(void*)NULL,(char*)NULL);
        }
    }
}
```

End Listings

Eco-C88 C Compiler with Cmore Debugger

Professionals prefer the Eco-C88 C compiler for ease of use and its powerful debugging features. Our "picky flag" gives you nine levels of lint-like error checking and makes debugging easy:

"I'm very impressed with the compiler, editor, and debugger. I've tried quite a few different compilers for the PC and have given up on all of the others in favor of yours... I've gotten to the point where I download C code from a DEC VAX/VMS system just to be able to compile it with the picky flag set at 9. It finds lots of things VMS totally ignores..."

JS, Oak Ridge, TN

The Eco-C88 compiler includes:

- A full-featured C compiler with 4 memory models (up to 1 meg of code and data) plus most ANSI enhancements.
- Without a doubt, the best error checking you can get. We catch bugs the others miss, making you much more productive.
- Cmore is a full-featured source code debugger, not some stripped-down version.
- Robust standard library with over 230 useful (no "fluff") functions, many of which are System V and ANSI compatible. Full source is available for only \$25.00 at time of order.
- CED, a fast, full screen, multiple-window program editor with on-line function help. You can compile, edit, and link from within CED.
- cc and mini-make utilities included that simplifies the most complex compiles.
- Users manual with over 150 program examples (not fragments) to illustrate how to use the library functions.
- Fast compiles producing fast code.

Our Guarantee: Try the Eco-C88 compiler for \$99.95. Use it for 30 days and if you are not completely satisfied, simply return it for a full refund. We are confident that once you've tried Eco-C88, you'll never use anything else. Call or write today!

Orders: **1-800-952-0472**
Info: **1-317-255-6476**



Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220

ECOSOFT

CIRCLE NO. 119 ON READER SERVICE CARD

Dr. Dobb's Journal Back Issues

**October 1987 #132 Volume XII,
Issue 10**

Stretching Apple Talk • Focus on
Forth: Unifying Dialects, Faster

**November 1987 #133 Volume XII,
Issue 11**

Special Graphics Issue • Tools for: 3-D
Mapping, Screen Management, Turbo C
Graphics

**February 1988 #136 Volume XIII
Issue 2**

Debugging on the 80386 • Making Serial
Links Work • New Product Review
Section • Languages: C, Forth, Pascal

March 1988 #137 Volume XIII Issue 3
Object-Oriented Design • Tools For
Handling: Binary Trees, Huge Arrays,
EGA Fonts • Reviews: Codeview, Turbo
Pascal 4.0

April 1988 #138 Volume XIII Issue 4
Creating A New AI Language • Combin-
ing Rules and Hypertext Links • Reviews:
Turbo Professional 4.0, Guidelines C++
Brief 2.0

Other issues are also available. Please
inquire.

To Order: Return this coupon with your
payment to: M&T Books, 501 Galveston Drive,
Redwood City, CA 94063. Or, CALL TOLL -
FREE 800-533-4372 (In CA 800-356-2002)

Please send the issues circled:

132 133 136 137 138

Price: 1 issue \$5.00; 2-5 issues \$4.50 each; 6 or
more \$4.00 each. There is a \$10 minimum for
charge orders.

Subtotal _____

Outside U.S. add \$.50 per issue _____

TOTAL _____

Name _____

Address _____

City _____ State _____ Zip _____

☐ Check enclosed. Make payable to M&T
Books.

Charge my ☐ Visa ☐ MC ☐ AmEx

Card No. _____ Exp. Date _____

Signature _____

3052G

A SIMPLE DECOMPILER

Listing One (Text begins on page 50.)

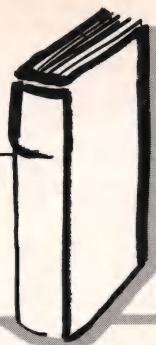
```

1  /*****
2  deparse.c
3
4  Decompiles an RPN expression into the original source code (or
5  a reasonable facsimile). Based on an algorithm by P.J. Brown in
6  'More on the Re-creation of Source Code from Reverse Polish'
7  Software - Practice and Experience, vol 7 pgs 545-551
8
9  William May
10 303 Ridgely Circle
11 Clinton, MA 01510
12
13 *****/
14
15 #include <types.h>
16 #include <ctype.h>
17
18 #define BUFSIZE 80
19 #define NIL (char *)0
20 #define TRUE 1
21 #define FALSE 0
22
23 enum {
24     NONARY_OP = 1,
25     UNARY_OP = 2,
26     BINARY_OP = 4
27 } lex_types;
28
29 /*-----
30 This table describes the operators that the code re-creating
31 algorithm will handle. The identifier is the internal representation
32 of an operator, the lexical type indicates the number of arguments to
33 the operator, and the prefixes and suffix supply transformations
34 of the operators.
35 -----*/
36 typedef struct decomp_row {
37     char ident;
38     short lex_type;
39     char *prefix_1;
40     char *prefix_2;
41     char *suffix;
42 } decomp_row;
43
44 /*-----
45 An example of the source recreation table setup.
46 Note that in real life the ids need not be printable
47 However, printable characters make the program much easier to test.
48 -----*/
49 static decomp_row table[] = {
50     {'+', UNARY_OP, "-", NIL, NIL},
51     {'+', BINARY_OP, NIL, "+", NIL},
52     {'-', BINARY_OP, NIL, "-", NIL},
53     {'/', BINARY_OP, NIL, "/", NIL},
54     {'*', BINARY_OP, NIL, "*", NIL},
55     {'^', BINARY_OP, NIL, "^", NIL},
56     {'(', UNARY_OP, "(", NIL, ")"},
57     {'$', UNARY_OP, "sum(", NIL, ")"},
58     {'=', BINARY_OP, "let ", "=", NIL},
59     {'', BINARY_OP, NIL, "", NIL}
60 };
61
62 #define TABLENUM (sizeof(table)/sizeof(decomp_row))
63
64 #ifdef DEBUG
65 #include <stdio.h>
66
67 /*-----
68 This is a simple test stub for the deparse function. It
69 reads a line from stdin, displays the line and the decompiled
70 result on stdout. Leave DEBUG undefined when compiling deparse as
71 a library.
72 -----*/
73 main(argc, argv)
74 int argc;
75 char *argv[];
76 {
77     char s[BUFSIZE]; /* input string */
78     char t[BUFSIZE]; /* output string */
79     int n;
80     void deparse();
81
82     printf("\n\nDecompiling test expressions:\n\n");
83
84     /* get a line. Note that buffer s still has linefeed in it */
85     while (fgets(s, BUFSIZE, stdin)) {
86         if (n = strlen(s)) {
87             s[n-1] = '\0';
88             printf("%s --> ", s);
89             deparse(s, t);
90             printf("%s\n", t);
91         }
92     }

```

(continued on page 70)

D ICTIONARY



A **Dictionary** is a collection of data pairs, much like an English language dictionary is a collection of term/definition pairs. The "term" entry is the key and the "definition" entry is the value of the pair, or association.

A Dictionary functions like a single-key database. This is how you create an object of class Dictionary with room for two key/value pairs:

```
Worker1 := new(Dictionary, 2);
```

Next fill the dictionary with strings as keys like "Name" and "Age" and the corresponding values, the string "Sam Jones" and the integer 22. You are free to mix data types.

```
Worker1["Name"] := "Sam Jones";  
Worker1["Age"] := 22;
```

Like a database, an Actor dictionary grows when you add to it more entries than you originally specified. Below, the dictionary grows from 2 to 4 as you add two more entries, or pairs:

```
Worker1["Department"] := #Engineering;  
Worker1["Reviews"] := #(85 73 98 94 100);
```

Mixing data types allows great flexibility. You can even incorporate the above dictionary, Worker1, as a value in an entry of another dictionary, as in Employees below:

```
Employees := new(Dictionary, 10);  
Employees["Sam"] := Worker1;
```

Access the data in a dictionary by specifying the appropriate keys. The following statement returns the array #(85 73 98 100):

```
AWorkersReview := Employees["Sam"]["Reviews"];
```

You can also retrieve data from dictionaries by enumerating over the dictionary and extracting particular entries. Here is how to create a separate "database" of employees working in the Engineering department:

```
Engineers := extract(Employees, {using(employee)  
employee["Department"] == #Engineering});
```

Another feature allows you to create a new database incorporating only a subset of the original's data. You do it by enumerating over a dictionary and collecting particular data about every entry:

```
MailingList := collect(Employees, {using(employee) employee["Address"]});
```

Dictionary objects are available in Actor as ready-made units of functionality. They can be modified or specialized for any application. However, as programmers, you are never involved in a dictionary's physical implementation or memory management.

Technical Specifications

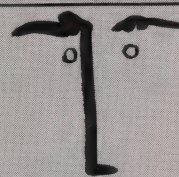
- Runs with Microsoft Windows 1.x, 2.0 and 386.
- Pure, single-inheritance object-oriented language, incrementally compiled.
- Dynamic linking to C, Pascal, Assembler, or Fortran libraries. Pass data in C structures.
- Pascal and C-like syntax.
- Programming tools: Browser, Inspector, Debugger, File Editor.
- Full access to MS-Windows systems calls, multi-tasking, and DDE.
- Fast device-independent graphics: lines, shapes, icons, cursors, bitmaps, metafiles, Turtle graphics, sample control language using YACC.
- 150 classes, 1500 functions, fully extensible.
- Window styles: tiled, overlapping, popup, child, edit, dialogs. Controls: list boxes, scroll bars, buttons, check boxes.
- Data structures: stacks, arrays, queues, lists, dictionaries, sets, sorting, hashing, intervals.

- AI support: frames, symbols, dictionaries, lists, symbolic programming, functional arguments. Parsing and lexical analysis YACC compatible.
- String manipulation: substring, concat, append, insert, remove, search.
- 643-page manual includes tutorial and reference.
- No license fees. Generates stand-alone applications.
- Fastest interactive OOL available.

Prices

Actor \$495 • Academic \$99 • Academic site license \$99 • Manuals for site license \$35 • **New!** Language Extension 1 \$99 • Shipping \$5 US, \$25 Int'l
Tech Support: Level 1 (20 phone calls) \$100 • Level 2 (Developer's package) \$250
Training Courses: 201 Intro. to Actor and OOP (2 days) \$495 • 202 Actor Developer's Workshop (3 days) \$995 • 203 Both 201 and 202 (1 week) \$1295

ACTOR[®]



Class of the Month

Vol. 1

No. 1

Actor: an interactive object-oriented programming environment running under Microsoft Windows.

Class: a template for data structures, or objects, with similar properties. These objects are tested, refined and reusable units of functionality you can snap into your Actor programs at any time.

The Whitewater Group[®]
Technology Innovation Center
906 University Place
Evanston, Illinois 60201

For more information call:
(312) 491-2370

Actor is a registered trademark of The Whitewater Group, Inc.



Serious programmers look for the Programmer's Power Pack—

the key to finding the
programming tools
and utilities that they
need to get ahead in
their work.

Programmer's Power
Pack is the one card
deck no serious pro-
grammer can live
without...and it's free!

Look for the
Programmer's Power
Pack in your
mailbox...it's coming
soon.

Or call Glynn
Mansfield at
415-366-3600 for more
information.

A SIMPLE DECOMPILER

Listing One (Listing continued, text begins on page 50.)

```

92     }
93
94     exit(0);
95 }
96 #endif
97
98 /*-----
99     deparse(): the only externally visible function, carries out deparsing
100                an RPN expression.
101     -----*/
102
103 void deparse(instr, outstr)
104 char *instr;
105 char *outstr;
106 {
107     char *restart; /* restart position after a forward scan */
108     int level; /* number of operands passed during a forward scan */
109     char *p, /* pointer to string to emit */
110         *prefix_1_for_elem(),
111         *prefix_2_for_elem(),
112         *suffix(),
113         *pop();
114     void init_stack(),
115         push();
116
117     init_stack();
118
119     /* make sure outstr is terminated */
120     *outstr = '\0';
121
122     /* scan the input string */
123     while (*instr) {
124
125         /* look for the next operand */
126         while (elem_is_operator(*instr)) {
127             if ((p = suffix(*instr)) != NIL)
128                 strcat(outstr, p);
129
130             advance_to_next_elem(&instr);
131
132             if (!(*instr))
133                 return; /* no more input, so quit */
134         }
135
136         /* found an operand, so scan forward for prefixes to this operand. */
137         level = 0;
138         restart = instr;
139
140         while (level >= 0) {
141             /* get the next token */
142             advance_to_next_elem(&instr);
143
144             /* have we reached the end of the input? */
145             if (!*instr)
146                 break;
147
148             /*
149              is the next token an operator or operand? If an operand then
150              update count of intervening operands (the level). If an operator
151              figure out if it results in a prefix to our operand (based on
152              the level)
153             */
154             if (elem_is_operand(*instr))
155                 level++;
156             else {
157                 if (elem_is_binary_op(*instr))
158                     --level;
159
160                 if (level == 0)
161                     if ((p = prefix_1_for_elem(*instr)) != NIL)
162                         push(p);
163
164                 if (level == -1)
165                     if ((p = prefix_2_for_elem(*instr)) != NIL)
166                         push(p);
167
168                 /* ... can be extended for more complex operators */
169             }
170         }
171
172         /*
173         unwind results of the forward scan by popping them off
174         the stack.
175         */
176         while (p = pop())
177             strcat(outstr, p);
178
179         /* forward scan complete, reset our scan point for another round */
180         instr = restart;
181     }
182 }

```



```

183      /*
184      now emit the operand itself. Note that operands will often
185      need conversions and formatting in real life, i.e. integer ->
186      string or floating point -> string conversions,
187      currency/date/time formatting, etc. Here we just append the
188      raw operand to the output string.
189      */
190      strncat(outstr, instr, 1);
191
192      advance_to_next_elem(&instr);
193  }
194  }
195
196  /*****
197  table handling utilities
198  *****/
199
200  /-----
201  elem_is_operator(): figure out if code is an operator
202                      if so, then return true
203  -----*/
204  static elem_is_operator(code)
205  char code;
206  {
207      decomp_row *row, *find_op();
208
209      if (row = find_op(code))
210          return TRUE;
211      else
212          return FALSE;
213  }
214
215  /-----
216  elem_is_operand(): figure out if code is an operand
217                    if so, then return true
218
219      In this example all operands are alphabetic or numeric characters.
220  -----*/
221  static elem_is_operand(code)
222  char code;
223  {
224      if (isalnum(code))
225          return TRUE;
226      else
227          return FALSE;
228  }
229
230  /-----
231  elem_is_binary_op(): figure out if code is a binary operator
232                      if so, then return true
233  -----*/
234  static elem_is_binary_op(code)
235  char code;
236  {
237      decomp_row *r, *find_op();
238
239      if (r = find_op(code))
240          return (r->lex_type & BINARY_OP);
241      else
242          return false;
243  }
244
245  /-----
246  prefix_1_for_elem(): get prefix 1 for the operator
247  -----*/
248  static char *prefix_1_for_elem(op)
249  char op;
250  {
251      decomp_row *r, *find_op();
252
253      if (r = find_op(op))
254          return (r->prefix_1);
255      else
256          return NIL;
257  }
258
259  /-----
260  prefix_2_for_elem(): get prefix 2 for the operator
261  -----*/
262  static char *prefix_2_for_elem(op)
263  char op;
264  {
265      decomp_row *r, *find_op();
266
267      if (r = find_op(op))
268          return (r->prefix_2);
269      else
270          return NIL;
271  }
272
273  /-----

```

(continued on next page)

DRAWBRIDGE

Drawbridge™ lets you create high-quality graphics displays for your applications, but saves you the tedious task of programming them.

That's right... just draw the picture on the screen using the mouse or the keyboard. At the press of a key Drawbridge automatically writes the source code calls to your graphics library to recreate the graphic. Copy the code into your application, compile it, and that's it!

Drawbridge is an object-oriented interactive editor that greatly increases your productivity when creating graphics displays.

Create high-quality graphics using features such as lines, ovals, arcs, polygons, fill patterns, color, and text fonts.

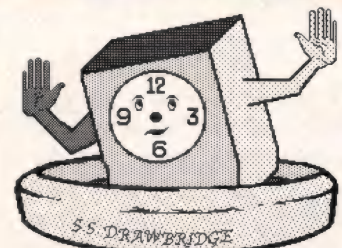
- Now for MetaWINDOW, Turbo C, and Microsoft C V5.0 graphics libraries
- C or Pascal source code
- 30-day money back guarantee

To order, or for more information, call:

(217) 359-1878

COURSEWARE APPLICATIONS, INC.

475 Devonshire Drive
Champaign, IL 61820



"It's a real timesaver!"

Versions are available now to generate Pascal or C language function calls for the MetaWINDOW™ graphics library (\$129) and C calls for the Turbo™ C and Microsoft™ C V5.0 graphics libraries (\$49). Specify language and library when ordering. A small restocking fee applies to returns.

CIRCLE NO. 110 ON READER SERVICE CARD

TOTAL CONTROL

with LMI FORTH™



For Programming Professionals:
an expanding family of
compatible, high-performance,
Forth-83 Standard compilers
for microcomputers

For Development:

Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

LMI Laboratory Microsystems Incorporated
 Post Office Box 10430, Marina del Rey, CA 90295
 Phone credit card orders to: (213) 306-7412

Overseas Distributors.

Germany: Forth-Systeme Angelika Flesch, Tittisee-Neustadt, 7651-1665
 UK: System Science Ltd., London, 01-248 0962
 France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
 Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
 Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

A SIMPLE DECOMPILER

Listing One

(Listing continued, text begins on page 50.)

```

274 suffix(): get the suffix of given code
275 -----*/
276 static char *suffix(code)
277 char code;
278 {
279     decomp_row *row, *find_op();
280
281     if (row = find_op(code))
282         return row->suffix;
283     else
284         return NIL;
285 }
286
287 /*-----
288 find_op(): finds the operator "op" in the decompiler table
289 if found, it returns a pointer to the row
290 if not, it returns NIL
291 -----*/
292 decomp_row *find_op(op)
293 char op;
294 {
295     int i;
296     decomp_row *row;
297
298     row = table;
299     for (i = 0; i < TABLENUM; i++, row++)
300         if (op == row->ident)
301             return row;
302
303     return NIL; /* no hit */
304 }
305
306 /*-----
307 advance_to_next_elem(): advance to next element
308 bump scan pointer as we move
309
310 the function assumes that all tokens are a single byte.
311 -----*/
312 static advance_to_next_elem(p)
313 char **p;
314 {
315     (*p)++;
316 }
317
318 /*-----
319 a simple stack implementation
320 -----*/
321
322 /* stack size in bytes */
323 #define STACKSIZE 40
324
325 /** a couple of globals for the stack */
326 static char **sp, **top;
327 static char stack[STACKSIZE];
328
329 /*-----
330 init_stack(): prepare the stack for use
331 -----*/
332 static void init_stack()
333 {
334     top = stack + STACKSIZE - 1;
335     sp = top + 1;
336 }
337
338 /*-----
339 pop(): pop a pointer sized value from the stack
340 -----*/
341 static char *pop()
342 {
343     if (sp <= top)
344         return (*sp++);
345     else
346         return NIL;
347 }
348
349 /*-----
350 push(): push a pointer sized value onto the stack
351 -----*/
352 static void push(p)
353 char *p;
354 {
355     *--sp = p;
356 }
357

```

End Listing

Why We're Betting a Million Lines of Code on the SAS/C™ Compiler.

At SAS Institute Inc., we've invested more than 10 years of research—and over a million lines of code—in the SAS® System, the world's leading data analysis software. So you can bet we left nothing to chance when we chose the C language for the next generation of our software.

We selected C for the portability it would bring to the SAS System, but weren't about to risk our code on just any mainframe C compiler. So we tried them all. When none could meet our exacting requirements, we created our own: the SAS/C compiler.

We Developed It.

Support It. Use It.

The SAS/C compiler set new standards for efficiency and technical quality, with:

- A source-level debugger that includes structure display, ABEND recovery, and debugger I/O exits for debugging specialized applications
- Reentrant object code
- Highly optimized generated code
- Use of standard IBM linkage conventions, with support for 31-bit addressing
- A CMS Rexx/TSO CLIST interface
- Support for signal handling including program checks and terminal interrupts, and non-standard signals such as timer interrupts and stack overflow
- Many built-in functions including string handling
- In-line assembler.

And when we combined these features with outstanding technical support and frequent updates—both provided free—software developers everywhere took notice. The SAS/C compiler is now the market leader, installed in hundreds of commercial firms and academic institutions.

Test It. Compare It.

FREE for 30 Days.

We're betting you've set the same high standards. That's why we'd like to send you the SAS/C compiler, under

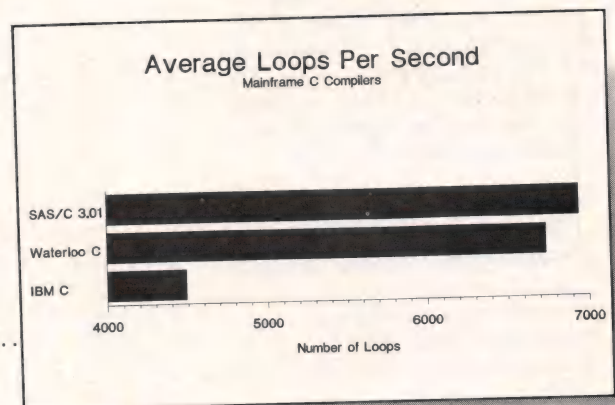
OS or CMS, for a free 30-day evaluation. We'll also send you a free copy of a leading benchmark program. Compare our compiler with any other. Odds are, you'll choose the SAS/C compiler.

Just mail the coupon below. Or call your Software Sales Representative at (919) 467-8000.



SAS Institute Inc.
SAS Circle ☐ Box 8000
Cary, NC 27512-8000
Phone (919) 467-8000
Fax (919) 469-3737

Using a C version of the Dhrystone benchmark, the latest SAS/C compiler release produces the fastest code among the top 3 mainframe compilers. It even tops our own previous release by 35%.



I'd like to put the SAS/C™ compiler to the test with a free 30-day trial, and my free copy of the Dhrystone benchmark program. Give me the details.

Please complete, or attach your business card.

Name _____ Title _____
Company _____
Address _____
City _____ State _____ ZIP _____
Telephone _____

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000,
Cary, NC, USA, 27512-8000

Promulate* your source codes from FORTRAN to C with PROMULA.FORTRAN.

**pro-mu-late vt a: to write source code in one language and compile it in another b: to learn a new language from another by translation c: to extend an application by sharing its information with a 4GL.*

PROMULA.FORTRAN is the FORTRAN to C translator for the IBM PC that has everything:

- Translation of FORTRAN-66, FORTRAN-77 and common extensions,
- Complete FORTRAN I/O including "large" FORMAT statements.
- Complex arithmetic including short and long complex numbers.
- FORTRAN-66 character-hiding in numeric variables and FORTRAN-77 character manipulation including variable-length strings.
- Multiple subroutine returns.
- Fractured FORTRAN code including token splits and other ambiguities.
- Optimized C code: reduced array subscripts and DO loop counters, multiple assignments, use of +=, -=, *=, /=, function prototypes to distinguish call-by-name and call-by-value, reduction of long integers to short when appropriate.
- Extensive runtime library: all FORTRAN intrinsic functions, a complete I/O system, two sets of complex functions, a virtual data management system, (object codes for Microsoft, Lattice, and Turbo C)
- Source code for library available with an archiver/full-screen editor.
- Automatic linkage to our 4GL, PROMULA: the virtual memory image formed on disk by a translated program can be used directly by PROMULA, thus providing access to report generation, graphics, edit and pick menus, extensive data management and other programs using the PROMULA system.
- PROMULA.PASCAL and BASIC currently under development.

- Translator: \$450
- Library Source: \$395
- Combined: \$745
- PROMULA: \$495
- Demo Disks \$10
- Manuals \$35

PROMULA Development Corp.
3820 N High Street., Ste. 301
Columbus, OH 43214
(614)263-5512

C CHEST

Listings One (Text begins on page 80.)

```
#include <malloc.h>

void      dfree      ( void *p );
void      *dmalloc   ( unsigned size );
void      *dcalloc   ( unsigned n, unsigned size );
int       malloc_checking ( int );
unsigned long memory_used ( void );

#ifdef DEBUG
#   define free(p)      dfree(p)
#   define malloc(s)    dmalloc(s)
#   define calloc(n,s) dcalloc(n,s)
#endif
```

End Listing One

Listing Two

Listing 1 -- dmalloc.c

```
1| #include <stdio.h>
2| #include <stdlib.h>      /* for prototypes only */
3| #include <malloc.h>      /* for prototypes only */
4| #include <string.h>      /* for prototypes only */
5| #include <dos.h>         /* SREGS definition */
6|
7| /* DMALLOC.C      Debugging versions of malloc and free for
8|  * the compact model.
9|  */
10|
11| #define MAXPTR 2048      /* max # of pointers to check */
12|
13| static char huge*  *Pointers = NULL;
14| static int         Nel = 0;      /*# elements in Pointers */
15| static int         Debugging_on = 0;
16| static unsigned long Total_mem = 0L;
17|
18| /*-----*/
19| int      malloc_checking ( int on );
20| void     *dmalloc        ( unsigned size );
21| void     *dcalloc        ( size_t n, size_t size );
22| void     dfree           ( char huge* p );
23| void     *find_in_table  ( char huge* key, int *found );
24| void     add_to_table    ( char huge* p );
25| int      remove_from_table ( char huge* p );
26| unsigned long memory_used ( void );
27| /*-----*/
28|
29| int      malloc_checking( on )
30| {
31|     /* Call this routine with a true argument to enable the
32|      * debugging features. Calling it with a false argument
33|      * frees all memory used for the checking.
34|      */
35|
36|     if( !(Debugging_on = on) )
37|     {
38|         Nel = 0;
39|         if( Pointers )
40|             free( Pointers );
41|         Pointers = NULL;
42|     }
43|     else
44|     {
45|         if( !Pointers ||
46|             !(Pointers = malloc(MAXPTR * sizeof(*Pointers))) )
47|         {
48|             fprintf(stderr, "No memory for pointer checking\n");
49|             return 0;
50|         }
51|     }
52|
53|     return 1;
54| }
55|
56| /*-----*/
57|
58| unsigned long      memory_used()
59| {
60|     return Total_mem;
61| }
62|
63| /*-----*/
64|
65| void     *dcalloc( n, size )
66| size_t n, size;
67| {
68|     void     *p;
69|
```

(continued on page 76)

UNLEASH YOUR 80386!

Your 80386-based PC should run two to three times as fast as your old AT. This speed-up is primarily due to the doubling of the clock speed from 8 to 16 MHz. The new MicroWay products discussed below take advantage of the real power of your 80386, which is actually 4 to 16 times that of the old AT! These new products take advantage of the 32 bit registers and data bus of the 80386 and the Weitek 1167 numeric coprocessor chip set. They include a family of MicroWay

80386 compilers that run in protected mode and numeric coprocessor cards that utilize the Weitek technology.

The benefits of our new technologies include:

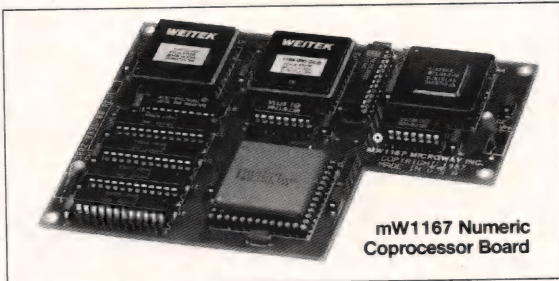
- An increase in addressable memory from 640K to 4 gigabytes using MS-DOS or Unix.
- A 12 fold increase in the speed of 32 bit integer arithmetic.
- A 4 to 16 fold increase in floating point

speed over the 80387/80287 numeric coprocessors.

Equally important, whichever MicroWay product you choose, you can be assured of the same excellent pre- and post-sales support that has made MicroWay the world leader in PC numerics and high performance PC upgrades. For more information, please call the Technical Support Department at

617-746-7341

After July 1988 call 508-746-7341



mW1167 Numeric Coprocessor Board

MicroWay® 80386 Support

MicroWay 80386 Compilers

NDP Fortran-386 and **NDP C-386** are globally optimizing 80386 native code compilers that support a number of Numeric Data Processors, including the 80287, 80387 and mW1167. They generate mainframe quality optimized code and are syntactically and operationally compatible to the Berkeley 4.2 Unix f77 and PCC compilers. MS-DOS specific extensions have been added where necessary to make it easy to port programs written with Microsoft C or Fortran and R/M Fortran.

The compilers are presently available in two formats: Microport Unix 5.3 or MS-DOS as extended by the Phar Lap Tools. MicroWay will port them to other 80386 operating systems such as OS/2 as the need arises and as 80386 versions become available.

The key to addressing more than 640 kbytes is the use of 32-bit integers to address arrays. NDP Fortran-386 generates 32-bit code which executes 3 to 8 times faster than the current generation of 16-bit compilers. There are three elements each of which contributes a factor of 2 to this speed increase: very efficient use of 80386 registers to store 32-bit entities, the use of inline 32-bit arithmetic instead of library calls, and a doubling in the effective utilization of the system data bus.

An example of the benefit of excellent code is a 32-bit matrix multiply. In this benchmark an NDP Fortran-386 program is run against the same program compiled with a 16-bit Fortran. Both programs were run on the same 80386 system. However, the 32-bit code ran 7.5 times faster than the 16-bit code, and 58.5 times faster than the 16-bit code executing on an IBM PC.

NDP Fortran-386™\$595
NDP C-386™\$595

MicroWay Numerics

The **mW1167™** is a MicroWay designed high speed numeric coprocessor that works with the 80386. It plugs into a 121 pin "Weitek" socket that is actually a super set of the 80387. This socket is available on a number of motherboards and accelerators including the AT&T 6386, Tandy 4000, Compaq 386/20, Hewlett Packard RS/20 and MicroWay Number Smasher 386. It combines the 64-bit Weitek 1163/64 floating point multiplier/adder with a Weitek/Intel designed "glue chip". The mW1167™ runs at 3.6 MegaWhetstones (compiled with NDP Fortran-386) which is a factor of 16 faster than an AT and 2 to 4 times faster than an 80387.

mW1167 16 MHz\$1495
mW1167 20 MHz\$1995

Monoputer™ - The INMOS T800-20 Transputer is a 32-bit computer on a chip that features a built-in floating point coprocessor. The T800 can be used to build arbitrarily large parallel processing machines. The Monoputer comes with either the 20 MHz T800 or the T414 (a T800 without the NDP) and includes 2 megabytes of processor memory. Transputer language support from MicroWay includes Occam, C, Fortran, Pascal and Prolog.

Monoputer T414-20 with 2 meg¹ ...\$1495
Monoputer T800-20 with 2 meg¹ ...\$1995

Quadputer™ can be purchased with 2, 3 or 4 transputers each of which has 1 or 4 megabytes of memory. Quadputers can be cabled together to build arbitrarily fast parallel processing systems that are as fast or faster than today's mainframes. A single T800 is as fast as an 80386/mW1167 combination!

Biputer™ T800/T414 with 2 meg¹ ...\$3495
Quadputer 4 T414-20 with 4 meg¹ ...\$6000

¹Includes Occam

80386 Multi-User Solutions

AT8™ - This intelligent serial controller series is designed to handle 4 to 16 users in a Xenix or Unix environment with as little as 3% degradation in speed. It has been tested and approved by Compaq, Intel, NCR, Zenith, and the Department of Defense for use in high performance 80286 and 80386 Xenix or Unix based multi-user systems.

AT4 - 4 users\$795
AT8 - 8 users\$995
AT16 - 16 users\$1295

Phar Lap™ created the first tools that make it possible to develop 80386 applications which run under MS-DOS yet take advantage of the full power of the 80386. These include an 80386 monitor/loader that runs the 80386 in protected linear address mode, an assembler, linker and debugger. These tools are required for the MS-DOS version of the MicroWay NDP Compilers.

Phar Lap Tools\$495

PC/AT ACCELERATORS

287Turbo-10 10 MHz\$450
287Turbo-12 12 MHz\$550
287TurboPlus-12 12 MHz\$629
FASTCACHE-286 9 MHz\$299
FASTCACHE-286 12 MHz\$399
SUPERCACHE-286\$499

MATH COPROCESSORS

80387-20 20 MHz\$895
80387-16 16 MHz\$495
80287-10 10 MHz\$349
80287-8 8 MHz\$259
80287-6 6 MHz\$179
8087-2 8 MHz\$154
8087 5 MHz\$99

MicroWay

The World Leader in PC Numerics

P.O. Box 79, Kingston, Mass. 02364 USA (617) 746-7341
32 High St., Kingston-Upon-Thames, U.K., 01-541-5466
St. Leonards, NSW, Australia 02-439-8400

Locate C Bugs before they Bite with PC-lint

PC-lint will analyze your C programs (one or many modules) and uncover glitches, bugs, quirks, and inconsistencies. It will catch subtle errors before they catch you. By examining multiple modules, PC-lint enjoys a perspective your compiler does not have.

"... a remarkable well thought-out product which will check for just about every conceivable coding error ... Its value increases with frequent use ... we confidently recommend PC-lint."

Andrew Binstock, The C Gazette

"PC-lint has everything going for it: flexibility, speed, good documentation, and a reasonable price. I exercised the product daily on a large, working, project to see if I could include it in my development tools. The answer is a definite YES."

*Stephen D. Cooper, Blue Notes
San Francisco PC Users Group*

NOW AVAILABLE
Shrouded Lint Source for use
on VAX/VMS, Versados, OS/9,
IBM VM/CMS - MVS, etc.
Requires only K & R C to
compile. Prices start at \$798.
Call for details.

Gimpel Software

3207 Hogarth Lane
Collegeville PA 19426
(215)584-4261

PRICE: \$139.00 first copy, \$100 each
additional, MC, VISA, COD, PA residents
add 6% sales tax, Outside USA add \$20.

Runs on MS-DOS - K&R C with ANSI
extensions - direct support for 12
MS-DOS C compilers including MS 5.0,
Turbo, Lattice, Desmet, Aztec
PC-lint is a trademark of Gimpel Software.

C CHEST

Listing Two (Listing continued, text begins on page 80.)

```

70|     if( p = dmalloc( size * n ) )
71|         memset( p, 0, size );
72|
73|     return p;
74| }
75|
76| /*-----*/
77|
78| void *dmalloc( size )
79| unsigned size;
80| {
81|     /* Debugging version of malloc. If debugging is enabled,
82|      * put the malloced pointer into the Pointers table before
83|      * returning it.
84|      */
85|
86|     void huge* p;
87|
88|     Total_mem += size;
89|
90|     if( !Debugging_on )
91|         return malloc( size );
92|
93|     if( !(p = (void huge*) malloc( size )) )
94|     {
95|         fprintf(stderr, "Malloc out of memory, returning NULL\n");
96|         return NULL;
97|     }
98|
99|     fprintf( stderr, "malloc(%u) = %p\n", size, p );
100|     add_to_table( p );
101|     return p;
102| }
103|
104| /*-----*/
105|
106| void dfree( p )
107| char huge* p;
108| {
109|     /* Debugging version of free(). If debugging is enabled,
110|      * check that the returned pointer is in the Pointers[]
111|      * table before allowing it to be freed(). The return-
112|      * address computation is far from portable, but it's
113|      * useful nonetheless.
114|      */
115|
116|     struct SREGS seg;
117|
118|     Total_mem -= _msize( p );
119|
120|     if( !Debugging_on )
121|     {
122|         free( p );
123|         return;
124|     }
125|
126|     if( !remove_from_table(p) )
127|     {
128|         segread( &seg );
129|         fprintf( stderr, "free() [Called from %04x:%04x]: BAD POINTER %p\n",
130|                 seg.cs, ((void(**)) ( &p ))[-1], p );
131|         exit( 1 );
132|     }
133|     else
134|     {
135|         free( p );
136|         fprintf( stderr, "free(%p) successful\n", p );
137|     }
138| }
139|
140| /*-----*/
141|
142| static void *find_in_table( key, found )
143| char huge* key;
144| int *found; /* set to 1 if found */
145| {
146|     /* Do a binary search in Pointers for key. If it's there
147|      * set *found to true and return a pointer to it; otherwise,
148|      * set *found to false and return a pointer to the place
149|      * in the table that it should go.
150|      */
151|
152|     char huge* *p; /* Pointer to middle element */
153|     char huge* *array; /* Pointer to base of array */
154|     int mid; /* Array index of middle element */
155|     int size;
156|
157|     *found = 0;
158|     if( !(size = Nel) )
159|         return Pointers;
160|
161|     array = Pointers;

```



```

162|
163| while( size > 0 )
164| {
165|     mid = size >> 1 ;
166|     p = array + mid ;
167|
168|     if ( key == *p )
169|     {
170|         *found = 1;
171|         return p;
172|     }
173|     else if ( key < *p )
174|     {
175|         size = mid ;
176|     }
177|     else
178|     {
179|         array = p + 1;
180|         size -- mid+1;
181|     }
182| }
183|
184| return (void *) ( *p > key ? p : p + 1 );
185| }
186|
187| /*-----*/
188|
189| static void add_to_table(p)
190| char huge* p;
191| {
192|     /* Add p to the Pointers table. If it's already there
193|      * print an error message.
194|      */
195|
196|     char huge* *tabp;
197|     int found;
198|
199|     tabp = find_in_table( p, &found );
200|
201|     if( found )
202|     {
203|         fprintf(stderr, "Malloc returned the same pointer twice!\n");
204|         exit( 1 );
205|     }
206|
207|     if( Nel >= MAXPTR )
208|     {
209|         fprintf(stderr,
210|             "Internal error [dmalloc()] too many pointers!\n");
211|         exit( 1 );
212|     }
213|
214|     if( tabp == &Pointers[Nel] )
215|         Pointers[Nel++] = p;
216|     else
217|     {
218|         ++Nel;
219|         memmove( tabp+1, tabp,
220|             (Nel - (tabp-Pointers)) * sizeof(*Pointers) );
221|         *tabp = p;
222|     }
223| }
224|
225| /*-----*/
226|
227| static int remove_from_table(p)
228| char huge *p;
229| {
230|     /* Remove p from the pointers table, return 0 if it's not
231|      * there, 1 otherwise.
232|      */
233|
234|     char huge* *tabp;
235|     int found;
236|
237|     tabp = find_in_table( p, &found );
238|
239|     if( !found )
240|         return 0;
241|
242|     --Nel;
243|     memmove( tabp, tabp+1,
244|         (Nel - (tabp-Pointers)) * sizeof(*Pointers) );
245|     return 1;
246| }
247|
248| /*-----*/
249|
250| #ifdef MAIN
251| main()
252| {
253|     void *p1, *p2, *p3, *p4, *p5;
254|     int i;
255|
256|     malloc_checking( 1 );

```

(continued on next page)

function libraries
disassemblers
compilers
text editors
text filters
communications support
text formatters
interpreters
bulletin boards
co-routines
compiler compilers
window packages
assemblers
games
tutorials
math packages
link editors
languages
cross compilers
pre-processors
function libraries
disassemblers
compilers
text editors

The C Users' Group
Library

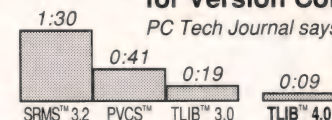
A Directory
of Public Domain
C Source Code

Send \$10
for Directory. Write
or call for more details
on over 100 volumes of
Public Domain C Source
Code.

The C Users' Group
PO Box 97
McPherson, KS 67460
(316) 241-1065

CIRCLE NO. 99 ON READER SERVICE CARD

for Version Control
PC Tech Journal says...



Times are to update a 45K library on a PC/XT. PVCS and TLIB 3.0 are from Sept 87 PC Tech Journal. SRMS and TLIB 4.0 are later versions.

TLIBTM is FASTEST!

"TLIB is a great system" *PC Tech Journal, March 88*
"TLIB...has my highest recommendation." *Ronny Richardson, Computer Shopper, August 87*

"If you've been putting off getting a revision control system, you no longer have an excuse." *The C Users Journal, February 88*

• **A Full-Featured System for Software Professionals**
Branching, for parallel development. Check-in/out locking. Keywords. Wildcard and list-of-file support; creates lists by scanning source code for includes. Can merge (reconcile) multiple simultaneous changes and undo intermediate revisions. Network and IBM 3363 optical disk support.

MS-DOS 2.x & 3.x **Just \$99.95 + \$5 s/h** Visa/MC
5 station LAN license \$299.95 + \$5 s/h
call for pricing on other network sizes

BURTON SYSTEMS SOFTWARE
PO Box 4156, Cary, NC 27519 (919) 469-3068

New!!

Graphics Library

BoosterGraphics v1.1

Now Available for Microsoft Quick Basic v 4.0
and most other Microsoft languages.
Also Borland Turbo Basic & Turbo C

**Create Graphics Effects & Graphics Based
user interfaces - Easily & Quickly, using CGA,
EGA, VGA, MCGA or Hercules adapters.**

Features Include:

Multipage Graphics, High Speed Drawing
Image Blitter Routines, Dot Matrix Printer Support
Instant pull-down menus ... And Much More

Price \$65.00 - 60 day money back guarantee
Assembly Source Code Available
No royalties - Free Technical Support

TO ORDER, please call us:

SUNCLOUD SOFTWARE, Inc.
101 West Ninth Street
Durango, Colorado 81301
(303) 247-0439

Add \$3.00 shipping/handling U.S., \$5.00 Canada.
Colorado residents please add 7% sales tax.

CIRCLE NO. 235 ON READER SERVICE CARD

Listing Two (Listing continued, text begins on page 80.)

```

257|
258|     p1 = dmalloc( 32767 );      ptab();
259|     p2 = dmalloc( 32767 );      ptab();
260|     p3 = dmalloc( 32767 );      ptab();
261|     p4 = dmalloc( 32767 );      ptab();
262|     p5 = dmalloc( 32767 );      ptab();
263|
264|     dfree( p1 );                 ptab();
265|     dfree( p2 );                 ptab();
266|     dfree( p3 );                 ptab();
267|     dfree( p4 );                 ptab();
268|     dfree( p5 );                 ptab();
269| }
270|
271| /*-----*/
272|
273| ptab()
274| {
275|     int i;
276|     printf("\tNel = %d\n", Nel );
277|     for( i = 0; i < Nel; ++i )
278|         printf("\tPointers[%d] = %p\n", i, Pointers[i] );
279| }
280| #endif

```

End Listings

Works with MS & TurboC

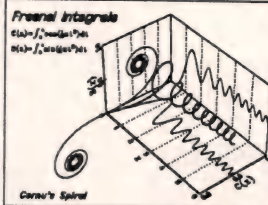
If you need the best source debuggers for MS-DOS or embedded microprocessor development with excellent support for C, PLM86, and assembly language contact:

SoftAdvances

10811 Washington Bl., Ste. 205 • Culver City, CA 90232 • (213) 559-7015

CIRCLE NO. 217 ON READER SERVICE CARD

GraphiC™
Publication
quality graphics
on your IBM® PC



- linear, log, polar plots
- bar charts, Smith charts
- 3D curves, 3D surfaces
- 6 curve types, 8 markers
- thick lines, panel fills
- 15 fonts, font editor
- 4096 x 3120 resolution
- zoom, pan, window plots
- high resolution printer & plotter dumps in color

Over 150 C and assembler routines for full control

\$395 with source code.

For personal use only.

MOST HARDWARE IS SUPPORTED
Scientific Endeavors Corporation

Route 4, Box 79 Kingston, TN 37763 (615) 376-4146

CIRCLE NO. 208 ON READER SERVICE CARD

VTEK™

DEC® VT100/VT52
and Tektronix®
4010, 4014, & 4105
Terminal Emulator

- 20 user-defined keys
- large scroll back buffer
- hardware 132 columns
- Kermit and XMODEM
- up to 800x600 screen resolution on EGAs
- zoom, pan, window plots
- "hot key" to DOS
- all VT100 keys, long and short breaks
- ANSI extensions to VT100 for multi-color text
- scrolling VT100 window on graphics screen
- convert files to .GEM & .PIC formats

\$150. Site and source code licenses available

B-EDIT™

Our new binary editor for programmers - \$29

ANNOUNCING



QuickMod

Better buckle up before getting behind the keyboard. You've never seen a Modula-2 system quite this fast!

QuickMod is a supercharged version of the critically-acclaimed Stony Brook Modula-2. You get 15000-line-per-minute compilation speed on 286's. And you don't have to link before running your program.

Look at what you get for just \$95.00:

Integrated compiler, text editor, make facility and windowing symbolic debugger • dynamic linking or EXE file option • imports Microsoft object modules • LONGINT • LONGREAL • 80x87 support or emulation • structured constants • array slices • Wirth edition 3 implementation

And, unlike most PC Modula-2 implementations, *QuickMod* is a true two-pass compiler that handles the exact scope and visibility rules specified by Nicklaus Wirth.

Stony Brook
INC.
SOFTWARE

For more information or to order, call or write:

Forest Road
Wilton, NH 03086

(603) 654-2525

The *QuickMod* language is compatible with Stony Brook Modula-2, so you can move up easily when you need more horsepower. That's a definite advantage, considering what reviews say about Stony Brook Modula-2:

"Stony Brook is a truly high-performance compiler that helps make Modula-2 the serious software developer's language of the 1990's... Everything Stony Brook has included is finely crafted and well documented."

— Warren Keuffel and William J. Schaller,
Computer Language, April 1988

"Stony Brook's is a terrific compiler... This experience with Stony Brook's Modula-2 sold me on the product."

— Kent Porter in **Dr. Dobbs Journal of Software Tools**, April, 1988.

"The Stony Brook Modula-2 compiler is remarkably impressive... Initial tests indicated this compiler is competitive with Microsoft C 5.0 and Borland's Turbo C 1.5 in compilation time and code efficiency."

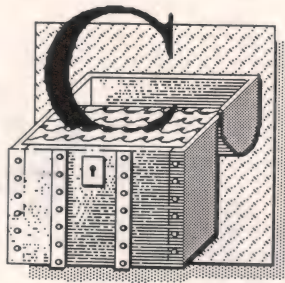
— Gene Hältiwanger, **Journal of Ada, Pascal and Modula-2**, March-April, 1988

QuickMod by Stony Brook Software for IBM PCs, PS/2s and compatibles:	\$95
Registered Stony Brook Modula-2 Users:	\$50
Stony Brook Modula-2 Development System for DOS or OS/2:	\$345
Both for only:	\$395

QuickMod gets you off the starting line FAST.

CIRCLE NO. 232 ON READER SERVICE CARD

Stalking the Wild Memory Allocator



One of the most difficult bugs to find in a program is a bad or uninitialized pointer, and one of the worst forms this bug can take is a pointer that has somehow insinuated itself into the memory pool used by *malloc()* and *free()*. The problem created by this bug stems from the way these routines interact with each other and with the operating system. The first time you call *malloc()*, it goes to the operating system to request that the memory space allocated to the current program be expanded. The routine then puts this extra memory in a local free list. This list is typically a linked list of headers, each of which contains a pointer to the next header in the list, and each of which is followed by a large block of available memory. A field in the header keeps track of the size of each block. The situation is illustrated in Figure 1, page 83.

When you request memory from *malloc()*, it carves a block of the desired size off the bottom of one of the available blocks in the free list, and the allocated chunk contains a header identical to the ones used in the free list. *malloc()* then returns a pointer to the area immediately below the header. In other words, the header can be examined by looking backward from the pointer returned from *malloc()*.

The problem here is that the header is used by *free()* when it puts memory back in the free list. If that header is corrupted, or if *free()*

by Allen Holub

is passed a pointer to an area of memory not preceded by a valid header [that is, a pointer that wasn't returned from a previous *malloc()* call], the continuity of the free list is destroyed. Similarly, strange things can happen if you pass a pointer to *free()* but continue to use it, or pass

the same pointer to *free()* twice. To be more specific, *free()* assumes that the header is both present and valid. Consequently, if a bad header is passed, the free list becomes corrupted.

If the size field is corrupted, the free list can contain overlapping memory blocks. That is, if the size of the first of two adjacent blocks is too large, the first block effectively overlaps the second. This means that *malloc()* eventually will allocate the same block of memory twice. The overlapping-memory problem is compounded by the fact that the header for the second block is now part of the data space in the first block. When that space is reallocated, your program can destroy the header used by the second block. In other words, the problem gradually propagates.

The free list can be corrupted in such a way that a block of memory that is outside of the program entirely (or a block that is in the program's code or static data space) is incorrectly linked. If this memory is returned from *malloc()*, you can inadvertently overwrite part of the system space or part of another program (such as *COMMAND.COM*) that is resident in memory. If you destroy *COMMAND.COM*, you'll just have to reboot. If you destroy the system space you can do things like erase the directory on your hard disk.

Note that this second problem happens only in the large-data model on the 8086 or on any machine that has a linear address space. Similarly, the problem is only critical in oper-

ating systems such as MS-DOS. In a real operating system, this sort of segmentation violation is detected and the offending program is aborted. If you're running under Unix, a file called *core* is created. This file can be examined with the *dbx* debugger to determine where the problem occurred. In MS-DOS, however, your program just dies a horrible death, or even worse, seems to work correctly but destroys your hard-disk directory as it runs.

Implementation

The listing this month describes a set of subroutines that I use to track down the problems just discussed. These subroutines are written for the compact model (compiled under Microsoft C), but should port to other compilers and models with little difficulty. The basic strategy is to put a subroutine around *malloc()* that keeps track of the returned pointers.

The outer subroutine, *dmalloc()*, is called by your program, and *dmalloc()* calls *malloc()*. Before returning the pointer *dmalloc()* remembers it in a table of pointers sorted numerically in ascending order. A version of *calloc()*, *dcalloc()*, does the same thing. A third subroutine, *dfree()*, surrounds *free()*. This third routine looks up the pointer in the table maintained by *dmalloc()*, and passes the pointer to *free()* only if the pointer appears in the table. (In this case, *dcalloc()* deletes the pointer from the table.) If the pointer is not present, *dfree()* prints an error message that tells you from where it was called (that is, it prints its own return address) and then calls *exit(1)*. If you're debugging with CodeView, the program terminates and the "Calls" window indicates the subroutine-calling sequence that got you into *dfree()*.

Two other support functions are available. The function

unsigned long memory__used()

returns the amount of memory currently in use (allocated with a *dmalloc()* or *dcalloc()* call, but not yet released by *dfree()*). The function

int malloc__checking(int on)

is used to enable or disable pointer checking at run time. The default is off, so the mechanism is not activated until *malloc__checking()* is called with a nonzero argument, at which time space for the pointer table is allocated. If called with a zero argument, the checking mechanism is disabled and the table space is freed. I generally control the presence or absence of the routines at compile time by using the macros shown in Listing 1, page 74. This way I can just call *malloc()* and *free()* in the usual way and the macros will do the mapping for me.

The debugging routines themselves are all in the file *dmalloc.c* (Listing Two, page 74). All the files included on lines 1 to 6 are supplied by Microsoft and most are ANSI include files. The *<dos.h>* file contains a define for the *SREGS* structure discussed later. The *MAXPTR* macro on line 11 determines the maximum number of pointers that can be active at any given moment (that is, pointers that have been returned from *dmalloc()*, but have not yet been passed to *dfree()*).

The *Pointers* array, declared on line 13, holds the pointers returned from *malloc()* calls. This array is actually a pointer itself, but will be initialized by *malloc__checking()* to point to the array. The array is sorted in ascending numeric order so that I can use a binary search to find a pointer. The declaration looks like this:

```
char huge* *Pointers ;
```

The *huge* keyword is used by the Microsoft compiler to define a true 32-bit pointer. The *huge* keyword modifies the star to its immediate right, so *Pointers* is a normal pointer to (an array of) *huge* pointers. *Huge* pointers must be used because of the way that normal far arithmetic is performed. The compiler makes the following assumptions about nor-

The Heap Expander™ version 2.0

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

still
\$59.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:

- Turbo C
- Microsoft C 4.0 and 5.0
- Turbo Pascal 3.0 and 4.0

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*Idaho residents add 5% sales tax
Foreign customers add \$4.00 for shipping and handling.

The Tool Makers

P.O. Box 8976
Moscow, Idaho 83843
208-883-4979



CIRCLE NO. 239 ON READER SERVICE CARD

ACCESS WORKSHEETS FROM



You can be accessing and creating worksheets and database files in one day.

WKS LIBRARY enables fast and reliable access to worksheets and databases. Programmers can easily use *wscanf()* and *wprintf()* to read and write worksheet rows. Over 50 functions. Cell values, formulas, macros, range-names, column widths, etc are all accessible.



SOFTWARE, INC.

WKS LIBRARY v. 2.0

- Reads & Writes WKS, WK1 and DBF Files
- 1-2-3 & Symphony Worksheet Compatible
- dBASEIII Database File Compatible
- Works with C Compilers from Microsoft, Lattice & Borland and Microsoft QuickBASIC 4
- DOS and XENIX
- C Source Included
- No Royalties For Executable Programs Distributed
- \$195

**Call (206) 828-4636 or
(800) 367-9882**

3055 112th Avenue N.E. Bellevue, WA 98004

DD 68T

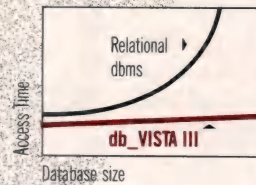
db_VISTA III™

Database Development System

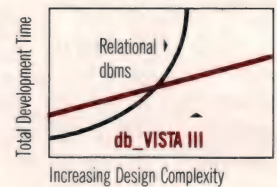


The db_VISTA III Difference

Continuous
High Performance



Consistent
Development Schedules



db_VISTA III takes your C programs as far as you dare to go.

Powerful applications require powerful tools. That's the idea behind db_VISTA III from Raima. db_VISTA III gives you powerful, high performance DBMS capabilities in any environment —VMS, ULTRIX, UNIX, XENIX, OS/2, DOS and Macintosh.

Our combination of a network model database, B-tree indexing, and SQL-based relational query, provide a superior system for data organization, manipulation and access.

db_VISTA III comes from a proven mainframe DBMS design. Because db_VISTA III is written in C for portability, we deliver this same power to minis and PCs.

**The best of both worlds...
network performance and
relational flexibility.**

db_VISTA III is a network model database. SQL-based db_QUERY is a rela-

tional interface. You can redesign existing databases with db_REWISE. An interface to Lotus 1-2-3 worksheets is also available.

For more information or to order call
800-327-2462 or (206)828-4636.
Find out how you can go as far as you dare.

The Raima db_VISTA III Database Development System Technical Information:

All components feature royalty-free run-time distribution, full source code availability and our commitment to customer service. That's why corporations like ARCO, AT&T, Hewlett-Packard, IBM, Northwestern Mutual Life, UNISYS and others use db_VISTA III.

1. db_VISTA: The High Performance DBMS

- ▶ Multi-user support
- ▶ Multiple database access.
- ▶ Record and file locking.
- ▶ Automatic database recovery.
- ▶ Transaction processing/logging.
- ▶ Timestamping.
- ▶ Database consistency check.
- ▶ Fast database access:
 - network model database
 - B-tree indexing
 - Virtual memory disk caching
- ▶ An easy-to-use interactive database access program
- ▶ File transfer program for importing/exporting ASCII.
- ▶ A Database Definition Language patterned after C.

2. db_QUERY: The SQL-based Query.

- ▶ Relational interface to db_VISTA databases.
- ▶ C-linkable or stand-alone usage.
- ▶ Embed queries or run ad-hoc queries.

3. db_REVERSE: The Database Restructure Program.

- ▶ Redesign your database easily.
- ▶ Converts all existing data to revised design.
- ▶ Send customized db_REVERSE to end user locations for easy upgrades.

4. WKS Library for Lotus 1-2-3.

- ▶ C-linkable interface to Lotus files.
- ▶ Reads and writes WKS, WK1 Worksheets and .dbf files.
- ▶ Control 1-2-3 from inside your C program.
- ▶ Use 1-2-3 as data entry and report screens.

Operating Systems:

- ▶ VMS, ULTRIX, UNIX, XENIX, OS/2, DOS and Macintosh.

C Compilers:

- ▶ VAX, UNIX, XENIX, Microsoft, Lattice, Turbo C, MPW, and LightspeedC.

LAN systems:

- ▶ LifeNet, NetWare, PC Network, 3Com, and other NET-BIOS compatible networks.

1-800-db-RAIMA (1-800-327-2462)

In Washington State call: (206) 828-4636
 In Texas call: (214) 239-3267
 In the U.K. call: (0992) 500919
 In Germany call: 07127/5244
 In Switzerland call: 01 725 04 10
 In France call: (1) 46 09 28 28
 In Belgium call: (02) 720.96.57

3055 112th Ave. N.E., Bellevue, WA 98004 U.S.A.
 Telex: 6503018237 MCI UW FAX: (206)828-3131



RAIMA™

C CHEST

(continued from page 81)

mal far pointers:

1. The object pointed to by a far pointer will not be larger than a segment (64K).
2. If two pointers are compared with each other, they are both assumed to point to the same array.

These rules mean that the segment portion of the 32-bit, far pointer must be loaded when the pointer is accessed, but it never has to be modified in an increment or used in a comparison. With the exception of indirect moves, the segment part of the pointer is ignored. On the other hand, huge arithmetic makes neither of the forgoing assumptions. Because an object can be larger than 64K, the segment portion of the pointer must be included in the arithmetic. I have to use huge pointers here because the pointers returned from *malloc()* will probably have different segment portions (in other words, the whole pointer must be examined).

The other variables declared on lines 14 to 16 are straightforward. *Nel* holds the number of pointers currently in *Pointers*. *Debugging_on* is true if the pointer checking has been activated by *malloc_checking()*. *Total_mem* keeps track of the total memory currently in use. Lines 19 to 26 are just prototypes for the functions in the file.

The *malloc_checking()* subroutine appears in lines 29 to 54. If

debugging is disabled, memory used for the *Pointers* array is freed (on line 40) and various flags are reset to their initial values; otherwise, the space for the *Pointers* array is allocated (on line 46). False is returned if the function can't get memory for the array. Note that nothing will happen if the *on* argument is true and debugging is already enabled. Similarly, nothing will happen if the *on* argument is false and debugging is already disabled.

The *dmalloc()* routine is on lines 78 to 102. If debugging is disabled, it simply modifies the *Total_mem* count and calls *malloc()* in the normal way. Otherwise, *dmalloc()* also puts the returned pointer into the table (with the *add_to_table()* call on line 100).

The *dfree()* routine (lines 106 to 137) works in a similar manner to *dmalloc()*, but it frees the memory and removes the pointer from the table rather than allocating it. The *segread()* call on line 128 (this is a routine supplied by Microsoft) reads the current values of the segment registers into the the SREGS structure. I'm using it to get the segment part of the return address to print the error message. Remember, this is a compact-model program, so the return address is a 16-bit near pointer. The return address itself is fetched with the somewhat strange looking expression on line 130:

```
( (void(**)( )) ( &p ) )[-1]
```

To see what what the return address is doing, consider the condi-

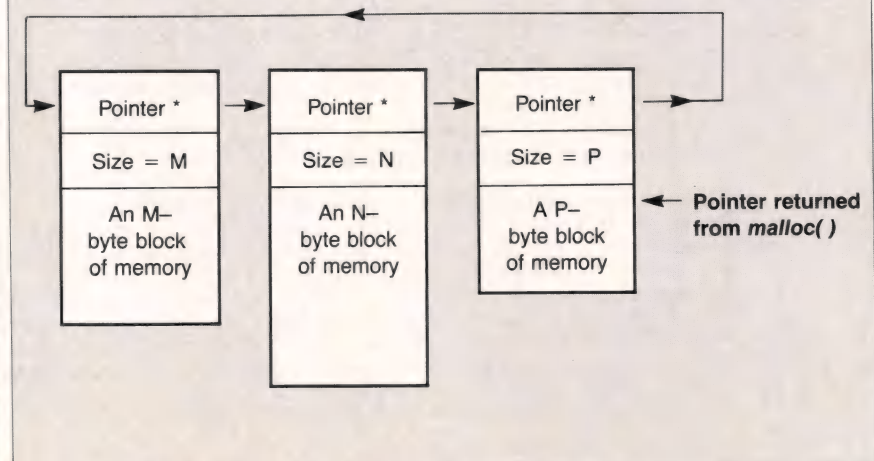


Figure 1: Free list organization

C CODE FOR THE PC

source code, of course

	Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	\$400
	CQL Query System (SQL retrievals plus windows)	\$325
	GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	\$325
	Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	\$300
NEW!	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	\$250
	Aspen Software PC Curses (System V compatible, extensive documentation)	\$250
	Greenleaf Data Windows (windows, menus, data entry, interactive form design)	\$250
	PforC++ (COM, database, file, user interface, & CRT C++ classes among others)	\$345
	Vitamin C (MacWindows)	\$200
NEW!	TurboTeX (TRIP certified; HP, PS, dot drivers; CM fonts; LaTeX)	\$170
	Essential resident C (TSRify C programs, DOS shared libraries)	\$165
	Essential C Utility Library (400 useful C functions)	\$160
	Essential Communications Library (C functions for RS-232-based communication systems)	\$160
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	\$150
	Greenleaf Functions (296 useful C functions, all DOS services)	\$150
	OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS)	\$150
	ME Version 2.0 (programmer's editor with C-like macro language by Magma Software)	\$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	\$135
	PC Curses Package (full System V, menu and data entry examples)	\$120
	CBTree (B+tree ISAM driver, multiple variable-length keys)	\$115
	Minix Operating System (U**x-like operating system, includes manual)	\$105
	PC/IP (CMU/MIT TCP/IP implementation for PCs)	\$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	\$100
	The Profiler (program execution profile tool)	\$100
	Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	\$100
	Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	\$100
	Wendin Operating System Construction Kit or PCNX, PCVMS O/S Shells	\$95
	C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules)	\$80
	Professional C Windows (windows and keyboard functions)	\$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	\$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	\$80
	WKS Library (C program interface to Lotus 1-2-3 program & files)	\$80
	Professional C Windows (lean & mean window and keyboard handler)	\$70
	Quincy (interactive C interpreter)	\$60
	EZASM (assembly language macros bridging C and MASM)	\$60
	PTree (parse tree management)	\$60
	HELP! (pop-up help system builder)	\$50
	Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	\$50
	Heap Expander (dynamic memory manager for expanded memory)	\$50
	Make (macros, all languages, built-in rules)	\$50
	Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	\$50
	Coder's Prolog (inference engine for use with C programs)	\$45
	C-Help (pop-up help for C programmers ... add your own notes)	\$40
	Biggerstaff's System Tools (multi-tasking window manager kit)	\$40
	PC-XINU (Comer's XINU operating system for PC)	\$35
	CLIPS (rule-based expert system generator, Version 4.1)	\$35
	TELE Kernel or TELE Windows (Ken Berry's multi-tasking kernel & window package)	\$30
	Clisp (Lisp interpreter with extensive internals documentation)	\$30
	Translate Rules to C (YACC-like function generator for rule-based systems)	\$30
	6-Pack of Editors (six public domain editors for use, study & hacking)	\$30
NEW!	Crunch Pack (a baker's dozen of file compression & expansion programs)	\$30
	ICON (string and list processing language, Version 6 and update)	\$25
	LEX (lexical analyzer generator)	\$25
	Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	\$25
	AutoTrace (program tracer and memory trasher catcher)	\$25
	C Compiler Torture Test (checks a C compiler against K & R)	\$20
	Benchmark Package (C compiler, PC hardware, and Unix system)	\$20
	TN3270 (remote login to IBM VM/CMS as a 3270 terminal on a 3274 controller)	\$20
	A68 (68000 cross-assembler)	\$20
	List-Pac (C functions for lists, stacks, and queues)	\$20
	XLT Macro Processor (general purpose text translator)	\$20
	Data	\$20
	WordCruncher (text retrieval & document analysis program)	\$275
	DNA Sequences (GenBank 52.0 including fast similarity search program)	\$150
	Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	\$60
	Webster's Second Dictionary (234,932 words)	\$60
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	\$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	\$30
	KST Fonts (13,200 characters in 139 mixed fonts: specify TeX or bitmap format)	\$30
	USNO Floppy Almanac (high-precision moon, sun, planet & star positions)	\$20
	NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	\$15
	U. S. Map (15,701 points of state boundaries)	\$15

The Austin Code Works

11100 Leafwood Lane

Austin, Texas 78750-3409 USA

acwlinfo@uunet.uu.net

Voice: (512) 258-0785

BBS: (512) 258-8831

FidoNet: 1:382/12

Free shipping on prepaid orders

For delivery in Texas add 7%

MasterCard/VISA

tion of the run-time stack when in *dfree()* (shown in Figure 2, page 85). The argument is pushed first, then the subroutine is called, pushing the return address. The address of the argument (&p) is the location of the low part of the 32-bit pointer (two words on the stack must be used to store the address). I've cast that address into a pointer to a function pointer because the return address is a function pointer (16 bits) as compared to a 32-bit data pointer. The [-1] references the cell at offset -1 (the one near pointer, or 2 bytes) from position of the argument. A glance at Figure 2 shows you that this is the return address. The expression could be restated as:

```
*( ((void(**))(&p)) - 1 )
```

This expression is more accurate, but harder to read.

The *find_in_table()* routine (lines 142 to 185) is a binary-search function that is passed a value for which to search (*key*) and returns two things. A pointer to the table entry that contains the key is normally returned, and, in this case, **found* is set to true. If the key isn't in the table, **found* is set to false and a pointer to where *key* should be is returned. For example, if the array holds the numbers 1, 3, 7, and

10 (in that order) and key is set to 5, a pointer to the 7 will be returned. This information is used to insert the value 5 in the table by first moving the 7 and 10 to the right one notch, and then by putting the 5 where the original 7 was found. Note that using a binary search is definitely worthwhile here. If the table is almost full (with 2K entries), the program takes at most 11 tries to find a given key (as compared to the 2,048 tries required for a worst-case linear search).

The *add_to_table()* subroutine (lines 189 to 223) adds a new pointer to the table. It prints an error message if the new entry is already in the table. Otherwise, this subroutine puts the new entry in the table, using *find_in_table()* (on line 199) to find out where the new entry belongs and inserts it on line 215 (if the entry is at the end of the array) or lines 219 to 221 (if the entry is in the middle of the array). Note that the Microsoft *memmove()* function must be used here rather than the standard *memcpy()* function because *memmove()* can handle overlapping regions while *memcpy()* can not. Otherwise, the routines are identical.

The *remove_from_table()* routine on lines 227 to 247 removes an entry from the table and closes up the end of the array to eliminate the hole. Zero is returned if the requested entry isn't there. Otherwise one is returned.

Extending the Routines

Though the routines just described have worked out quite well in practice, they could be extended farther by modifying the *Pointers* array so that it contains the entire header as well as the pointer. You could then test the entire header for consistency before freeing the memory. You could also save only the pointer and the block size rather than the whole header.

A second extension is to modify *malloc_checking()* so that the size of the *Pointers* array is passed as an argument. Then use this size on line 46 rather than the *MAXPTR* macro.

Another extension, which is practical only if you purchase the library source code from Microsoft, is to replace the standard *malloc()* and

V-LIB

the user interface library for C

V-LIB is a comprehensive, easy to use library of over 150 custom C functions for building sophisticated PC applications.

Windows • overlapping, tiled, built-in window management

Menus • vertical, horizontal pull-down, popup, arrays

Forms • full screen or popup data entry with multiple field types

Pop Ups • messages, prompts, selection lists, scroll bars

Formatted Screen Output

Full Editing Input

High Speed Display

... and much, much more!

Give your programs a professional look! Develop applications faster!

Compilers supported: Microsoft C and Quick C, Borland Turbo C, Lattice C, and Datalight C.

All memory models supported.

Demo disk with usable sample programs, source, and reference card . . . \$10

Library with reference card and 280 page programmer's manual . . . \$99

Library (as above) with full source code . . . \$149

Prices include shipping. California residents please add 7% sales tax.

No royalties. Site license available.

Call or write:

Pathfinder Associates

291 Madrone Avenue
Santa Clara, CA 95051
(408) 984-2256

Visa and MasterCard accepted.

BBS: (408) 246-0164 (1200/2400 N-8-1)

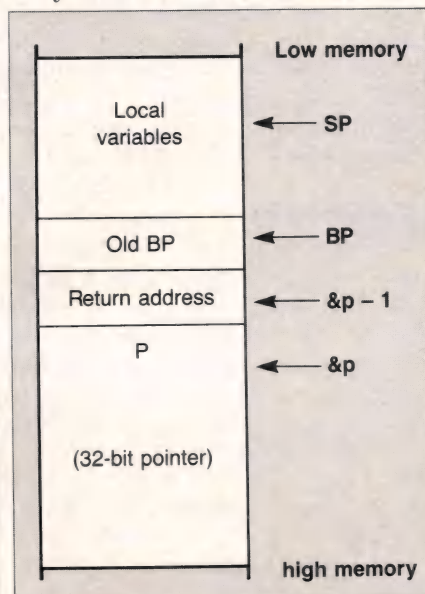


Figure 2: The stack after *defree()* has been entered

*You can
go home again!*

The MKS Toolkit brings UNIX™ to the DOS environment.

Only the MKS Toolkit provides users of DOS with:

- over 120 utilities with UNIX™ functionality;
- the versatility of the Korn shell;
- MKS Awk: the little language with the big power;
- MKS Vi: the lightning-fast full-screen editor.

ANNOUNCING . . .

Version 2.3 of the MKS Toolkit has added **spell**, **yacc**, **diff3**, and others, **PLUS** improved performance from the entire package.

**Buy the software that
"exceeds your expectations!"**

\$169.00

Updates to existing licences:

\$45.00

Within continental U.S.A. call

1-800-265-2797

Elsewhere:

1-519-884-2251

Mortice Kern Systems Inc.
is committed to POSIX standards.

MKS is a trademark of
Mortice Kern Systems Inc.



MKS Toolkit

C CHEST

(continued from page 85)

`free()` in the run-time library with the debugging versions. This way you don't have to worry about whether you've included the mapping macros everywhere you used one of the memory-management routines. You'll have to modify the sources a little to make this change.

The files of interest are `../heap/fmalloc.asm` and `../heap/nmalloc.asm`. The `malloc()` calls are mapped to `_fmalloc()` or `_nmalloc()` by statements in these files, and `free()` is mapped in a similar manner. To supply debugging versions, remove the following two lines from these two files:

```
labelP <PUBLIC,free> labelP  
<PUBLIC,malloc>
```

(Note that these lines are not adjacent in the source files.) Then rebuild the libraries. Next, go into `dmalloc.c` and rename `dmalloc()` to `malloc()` and `dfree()` to `free()`. Also change all `malloc()` references to `_fmalloc()` or `_nmalloc()` as appropriate. Map `free()` to `_ffree()` or to `_nfree()` as appropriate. Finally, compile `dmalloc.c` and link it into the run-time library.

Bibliography

The source code for `malloc()` and `free()` is presented in Kernighan and Ritchie, *The C Programming Language* (Englewood Cliffs: Prentice-Hall, 1978), pp. 173-177. Note that, for mysterious reasons, `malloc()` is called `alloc()`.

Availability

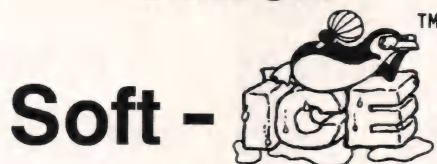
All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063, or call 415-366-3600, ext. 221. Please specify the issue number and format (MS-DOS), Macintosh, Kaypro).

DDJ

(Listings begin on page 74.)

Vote for your favorite feature/article.
Circle Reader Service No. 4.

SERIOUS DEBUGGING AT A REASONABLE PRICE



All the speed and power of a hardware-assisted
debugger at a software price

Hardware-level break points

REAL-TIME break points on memory locations, memory ranges, execution, I/O ports, hardware and software interrupts. More powerful break points than ANY software-only debugger on the market. Soft-ICE gives you the power of an in-circuit emulator on your desk.

Break out of hung programs

With a keystroke - no external switch necessary. Even with interrupts disabled.

Breaks the 640K barrier

Soft-ICE uses ZERO bytes of memory in the first 1MB of address space. This is especially useful for those subtle bugs that change when the starting address of your code changes. With Soft-ICE your code executes at the same address whether the debugger is loaded or not.

Works with your favorite debugger

Soft-ICE can be used as a stand-alone debugger or it can add its powerful break points to the software debugger you already use. You can continue to use your favorite debugger, but add Soft-ICE's powerful break point capabilities to handle the tough problems that your debugger is not suited for. For example you suspect a BIOS call is over-writing your code, but you don't know where. You pop up Soft-ICE, set a range break point across your code segments. If code is overwritten during the BIOS call Soft-ICE comes up with the location that was over-written. And this is all done at full 80386 speed!

Solve tough systems problems too

Soft-ICE is ideal for debugging TSRs, interrupt handlers, self booting programs, DOS loadable device drivers, non-DOS operating systems and debugging within DOS & BIOS. Soft-ICE is also great for firmware development because Soft-ICE's break points work in ROM.

How Soft-ICE Works

Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to provide real-time hardware-level break points.

"It is a unique, effective solution to a wide range of critical debugging problems."

COMPUTER LANGUAGE -- April 1988

NEW! NEW! NEW! NEW!

RUN CODEVIEW IN ONLY 8K™

MagicCV



CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 microprocessor to run CodeView in extended memory. This allows MagicCV to run CodeView using less than 8K of conventional memory on your 80386 PC.

Don't let 640K be your limit!

If you are closing in on the 640K limit and would like the power of CodeView, MagicCV is for you.

Don't let the debugger change the bug!

Even if you're not closing in on the 640K limit, running CodeView with MagicCV makes your debugging environment much closer to the end user's program environment. You can use CodeView to locate subtle bugs that only occur when there is plenty of free memory, or those difficult bugs that only occur when your program is running with a couple of TSRs loaded.

How MagicCV works

MagicCV uses the 80386 to create a separate virtual machine for CodeView. MagicCV uses between 4K & 8K of conventional memory as a bridge between the DOS environment and CodeView.

MagicCV is easy to use

If you are a CodeView user, you already know how to use MagicCV too. Everything is automatic. You can be solving problems within 10 minutes of opening the package.

SAVE \$86

MagicCV	\$199
Soft-ICE	\$386
MagicCV & Soft-ICE	\$499

CALL TODAY
(603) 888 - 2386

30 day money-back guarantee
Visa and Master Card accepted

NU-MEGA TECHNOLOGIES

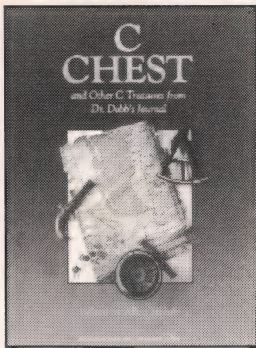
PO Box 7607
Nashua, NH 03060-7607

Both require 80386 AT compatible or IBM PS/2 Model 80.
MagicCV requires at least 384K of extended memory.
CodeView is a trademark of Microsoft Corporation.

MagicCV with Soft-ICE

Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere. As an extra bonus, by ordering both MagicCV and Soft-ICE together you save \$86.

EVERYTHING



C CHEST AND OTHER C TREASURES

by Allen Holub

This comprehensive anthology contains the popular "C Chest" columns by Allen Holub from *Dr. Dobbs's Journal of Software Tools*.

For the novice and experienced C programmer alike, **C CHEST AND OTHER TREASURES** will prove to be an invaluable resource, providing hours worth of information to be learned and applied.

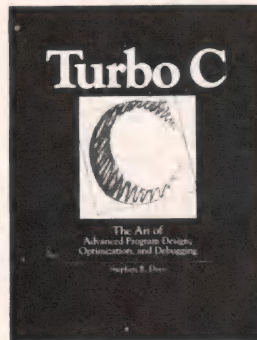
Just some of the many topics detailed are: pipes, wild-card expansion, and quoted arguments; sorting routines; command line processing; queues and bit maps; ls, make and other utilities; expression parsing; hyphenation; an Fget that edits; redirection; accessing IBM video display memory; trees; and AVL tree database package; directory transversal; sets; shrinking and .EXE file images.

The anthology also includes several information-packed articles written by well known C experts. Learn from the experts about a flexible program that allows you to find the minima of complex, multiple dimension equations; cubic-spline routines that provide an efficient way to do a more restrictive curve-fitting application; an fgrep program that resurrects a very efficient finite-state-machine based algorithm that can be used in any pattern-matching algorithm, and more!

C CHEST AND OTHER C TREASURES provides a collection of useful subroutines and practical programs written in C, and are available on disk with full source code.

Book & Disk (MS-DOS)
Book only

Item #49-6 \$39.95
Item #40-2 \$24.95



TURBO C: THE ART OF ADVANCED PROGRAM DESIGN, OPTIMIZATION AND DEBUGGING

by Stephen R. Davis

Packed with useful example programs, this book details the techniques necessary to skillfully program, optimize and debug in Turbo C. Every topic and Turbo C feature is fully demonstrated in Turbo C source code examples.

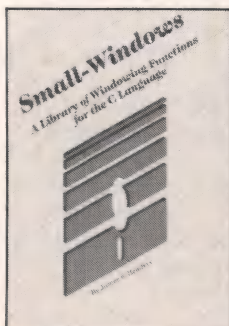
Starting with an overview of the C language, the author advances to topics such as pointers, direct screen I/O; inline statements in Turbo C; and how to intercept and redirect BIOS calls, all of which are demonstrated in a RAM resident pop-up program written in Turbo C.

Fully outlined are the differences between Unix C and Turbo C; the transition from Turbo Pascal to Turbo C; and the superset of K&R C features implemented in the proposed ANSI C standard.

Whether you are a C programmer interested in investigating this exciting new dialect of the language or a Turbo Pascal programmer who is curious to learn more of this C language, **TURBO C** is must reading!

Book & Disk (MS-DOS)
Book only

Item #45-3 \$39.95
Item #38-0 \$24.95



SMALL-WINDOWS: A LIBRARY OF WINDOWING FUNCTIONS FOR THE C LANGUAGE

by James E. Hendrix

SMALL-WINDOWS is an extensive library of C language functions for creating and manipulating display windows. The manual and disk package contains 41 windowing functions that allow you to clean, frame, move, hide, show, scroll, push and pop windows. Also included are 18 video functions written in assembly language, and menu functions that support both static and pop-up menus.

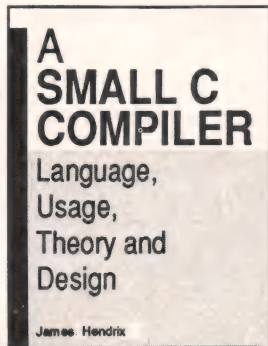
A file directory illustrates the use of window menu functions and provides file selection, renaming and deletion capability. Two test programs are provided as examples to show you how to use the library and the window, menu, and directory functions.

SMALL-WINDOWS is available for MS-DOS systems, and Microsoft C Versions 4.0/5.0, Turbo C 1.5, Small-C and Lattice C 3.1 compilers. Documentation and full C source code included.

Manual & Disk (MS-DOS)
(Microsoft C, Small-C, Lattice C, or Turbo C Compiler)

Item #35-6 \$29.95

THERE IS TO C!



A SMALL-C COMPILER: LANGUAGE, USAGE, THEORY AND DESIGN

by James E. Hendrix

For anyone using or considering Small-C, **A SMALL C COMPILER** provides valuable information about the language and its compiler.

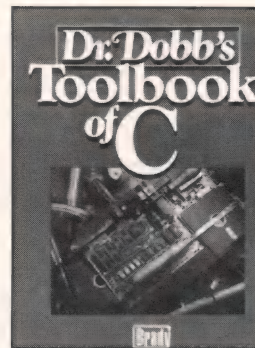
This book contains a full presentation of the design and operation theory of the Small-C compiler and programming language. In addition to a full, working Small-C compiler, this book provides an excellent example for learning basic compiler theory. Here is a real compiler that is easy enough to be understood and modified by computer science students, and may be transformed into a cross-compiler or completely ported to other processors.

Features include code optimizing, internal use of pseudo code, upward compatibility with full C, recursive descent parsing, a one pass algorithm, and the generation of assembly language code. You'll even learn how the compiler can be used to generate a new version of itself! No other compiler available to the public has ever been so thoroughly documented.

The accompanying diskette includes an executable compiler, fully documented source code and many sample programs. A Microsoft or IBM Macro Assembler is required.

Book & Disk (MS-DOS)
Book only

Item #97-6 \$38.95
Item #88-7 \$23.95



DR. DOBB'S TOOLBOOK OF C

by The Editors of
Dr. Dobb's Journal
of Software Tools

More than 700 pages of the best of C articles and source code from **Dr. Dobb's Journal of Software Tools** in a single book! Not just a compilation of reprints, this comprehensive volume contains new materials by various C experts as well as updates and revisions of some older articles.

The essays and articles contained within this virtual encyclopedia of information were designed to give the professional programmer a deeper understanding of C by addressing real world programming problems, and how to use C to its fullest.

Some of the highlights include an entire C compiler with support routines, versions of various utility programs such as Grep and a C program cross-referencer.

DR. DOBB'S TOOLBOOK OF C is an invaluable resource that you'll turn to again and again for an in-depth appreciation of C.

Book only

Item #615-3 \$29.95

To Order: Return this coupon with your payment to: M&T Books, 501 Galveston Dr., Redwood City, CA 94063
Or, **CALL TOLL-FREE 800-533-4372** Mon-Fri 8AM-5PM PST (In CA, Call 800-356-2002)

Name _____
Address _____
City _____ State _____ Zip _____

Item#	Description	Price

Subtotal _____
CA residents add sales tax _____ % _____
Add \$2.95 per item for shipping _____
Total _____

For *Small Windows*, indicate:

- ☐ Microsoft version 4.0/5.0 compiler
- ☐ Small-C compiler
- ☐ Lattice C 3.1 compiler
- ☐ Turbo C 1.5 compiler

☐ Check enclosed. Make payable to M&T Publishing.

Charge my: ☐ Visa ☐ MC ☐ AmEx
Card No. _____ Exp.Date _____
Signature _____

3052A

Real-Time Tidbits and some Words on Floating Point

“Using a real-time operating system to encase your application is like wearing armor into battle. The armored knight was better protected than an unarmored warrior. But the extra weight he was carrying also made him slower and less agile.” So writes Charles H. Small in his article “Real-Time Operating Systems” (*EDN*, January 7, 1988). Speaking further on reentrancy, Mr. Small reminds his readers that “Some languages, such as Forth, produce inherently reentrant code. Other languages require discipline on the part of the programmer and a special compiler that produces ROMable code.” (It’s easy to keep Forth code reentrant. Just keep arguments and parameters on the stack or in *USER* variables, and keep strings at *PAD*.)

Mr. Small then goes into a thoughtful and detailed description of Forth Inc.’s polyFORTH multitasker. This much copied round-robin multitasker is characterized by its *PAUSE* command. *PAUSE* is every task’s entry into a circularly linked queue, which eventually surfaces to execute the next awake task. (Tasks are generally asleep until awoken by an interrupt.) Each task retains control of the CPU for as long as it wishes, handing it off to the next task when convenient. This is apparently also the model for the Laxen/Perry public-domain F83, although nobody, but nobody, knows as much about round-robin multitasking as FORTH Inc. does. I should also like to point

by Martin Tracy

out to speed freaks everywhere that the time required to put one task to sleep, traverse the round robin, and wake the next task (that is, the task-switching time) on a DSP polyFORTH running on a TMS32020 chip is less than 2 microseconds.

Curiously, an article by Max Schin-



dler called “Toward Faster and Portable Real-Time Operating Systems” in *Electronic Design* (January 21, 1988) does not even mention Forth, even though this same issue contains “Combine Forth with Other Tools for Rapid Software Development” by W.H. Payne.

The Proceedings of the 1987 Rochester Forth Conference (reviewed here in the October 1987 issue of *DDJ*) are now available as vol. 5, no. 1, of the *Journal of Forth Application and Research* (*JFAR*). The 256 pages contain more than 50 papers on comparative computer architectures and many other topics. The subscription rate is \$50 per year and back issues are \$15 each. Call 716-254-0621 and ask for Forth.

Drs. Lou Odette and William B. Dress, well known to both the Forth and the AI communities, sent in a copy of their jointly written “Engineering Intelligence into Real-Time Applications” from the November 1987 *Expert Systems* (vol. 4, no. 4). The article focuses on the performance of knowledge-based, intermediate-size computer systems, such as you might expect to find in self-testing and self-calibrating machinery for manufacturing or monitoring. The authors present methodology to “effect the transfer of knowledge-base technology to real-time systems while meeting the constraints of embedded applications.” They present an interesting graph of hardware and software choices, which I have attempted to duplicate in Figure 1, page 91.

The authors’ approach is to construct a compiler for OPS/Prolog that

emits compact code for a highly portable, threaded-code intermediate language (Forth). The code, in turn, is used to implement the abstract machine underlying OPS5 and Prolog. Its first real-time application will be to control the Microgravity Vestibular Investigation, designed for experiments in space motion sickness aboard the Spacelab (currently scheduled for April 1991). The bibliography presented at the end of the article is worth an article in itself.

Call for Presentations

Get out your calendars and mark off November 18 and 19 for the 1988 Forth Convention at the Grand Hotel in Anaheim, California. This year’s theme is real-time, real-world programming. If you ever needed an excuse to bring your family to see Disneyland, this is it.

The convention is sponsored by the Forth Interest Group (FIG). We need presentations (not papers) on real-time programming and related topics, such as language-oriented RISC machines, working neural nets, Forth in aerospace, et cetera. Just send a short abstract to FIG, P.O. Box 8231, San Jose, CA 95155, or call 408-277-0668. We also need volunteers to chair and participate in panel discussions.

Come meet your vendor, or find your local FIG group, or hear Chuck Moore’s fireside chat, or watch a contest to find the world’s fastest programmer, or... Don’t miss this exciting event! I’ll remind you about it again later.

The Third ANSI X3J14 Meeting

The third meeting of the X3J14 ANSI Forth Technical Committee was held at Redondo Beach, California, on February 10–12, 1988. You can download Ray Duncan’s brief report on the meeting from almost any Forth BBS (try the ECFB or GENie).

But first, a brief review of the Standard process so far.

According to Ray, "At its first meeting, the X3J14 Technical Committee adopted the text of the Forth-83 Standard (less the Experimental Proposals) as its working, or BASIS, document. The BASIS will evolve into the draft proposed American National Standard (dpANS) document as the TC [Technical Committee] ratifies Technical Proposals which delete, amend, or enlarge the BASIS."

At the second meeting, the only significant change to the BASIS, other than reformatting it, was to relax the restriction on 16-bit stacks and addresses and allow Standard systems with other word sizes. The technical highlights of the third meeting included:

1. Removing the requirement for floored division while still making it available for existing programs. In effect, division by negative numbers yields an implementation-dependent quotient and remainder, which satisfy the equation that quotient times dividend plus remainder equals divisor.
2. Adding *NIP* and *TUCK* to the Controlled Reference Word Set.
3. Adding *EVAL*, a word that allows the interpretation of arbitrary strings. (Remember, you saw it first here.) To quote Ray Duncan again, "When *EVAL* is available, words that redirect the input stream can be

readily ported from one system to another, and new interpreters can be easily built in an implementation-independent manner."

Although not yet passed, and much to everyone's surprise, a Technical Proposal for a minimum extension set of floating-point operators was scheduled for fine-tuning and possible action at a later meeting. So, in honor of this commitment, I will direct the technical content of this month's column accordingly.

Floating Point

Conservative Forth programmers disdain floating-point for higher-performance, fixed-point arithmetic. "If you need to use floating-point arithmetic," they say, "then you don't understand the problem." Nevertheless, virtually all major Forth vendors now offer a floating-point packaged extension. Most of these extensions closely follow the hardware or firmware floating-point support on the host system, such as the 8087 coprocessor or the Macintosh SANE interface. Software-only, floating-point implementations usually follow the guidelines in "The FVG Standard Floating-Point Extension" by Duncan and Tracy (DDJ, September 1984).

Virtually all floating-point (FP) extensions provide the four functions named *F+*, *F-*, *F**, and *F/*. But what stack are the arguments on? Hard-

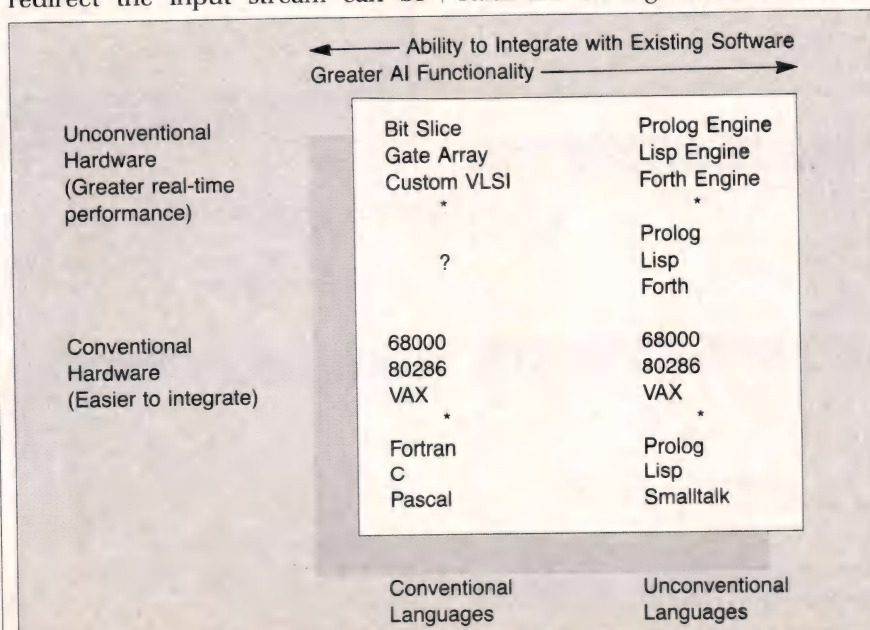


Figure 1: Hardware and software choices

COMBINE THE
RAW POWER OF FORTH
WITH THE CONVENIENCE
OF CONVENTIONAL LANGUAGES

HS / FORTH

Yes, Forth gives you total control of your computer, but only HS/FORTH gives you implemented functionality so you aren't left hanging with "great possibilities" (and lots of work!) With over 1500 functions you are almost done before you start!

WELCOME TO HS/FORTH, where megabyte programs compile at 10,000 lines per minute, and execute faster than ones built in 64k limited systems. Then use AUTOOPT to reach within a few percent of full assembler performance — not a native code compiler linking only simple code primitives, but a full recursive descent optimizer with almost all of HS/FORTH as a useable resource. Both optimizer and assembler can create independent programs as well as code primitives. The metacompiler creates threaded systems from a few hundred bytes to as large as required, and can produce ANY threading scheme. And the entire system can be saved, sealed, or turnkeyed for distribution either on disk or in ROM (with or without BIOS).

HS/FORTH is a first class application development and implementation system. You can exploit all of DOS (commands, functions, even shelled programs) and link to .OBJ and .LIB files meant for other languages *interactively!*

I/O is easier than in Pascal or Basic, but much more powerful — whether you need parsing, formatting, or random access. Send display output through DOS, BIOS, or direct to video memory. Windows organize both text and graphics display, and greatly enhance both time slice and round robin multitasking utilities. Math facilities include both software and hardware floating point plus an 18 digit integer (finance) extension and fast arrays for all data types. Hardware floating point covers the full range of trig, hyper and transcendental math including complex.

Undeniably the most flexible & complete Forth system available, at any price, with no expensive extras to buy later. Compiles 79 & 83 standard programs. Distribute metacompiled tools, or turnkeyed applications royalty free.

HS/FORTH (complete system): \$395.
HS/FORTH: Tutorial & Ref (500 pg) \$ 59.
Forth: Text & Reference (500 pg) \$ 22.
HS/FORTH Glossary \$ 10.
GIGA FORTH Option (Beta release) \$245.
(Native Mode from SOFTMILLS, INC.)



Visa

Mastercard



HARVARD SOFTWARES

PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

CIRCLE NO. 137 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP

helps save time, money, and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

OPTASM Optimizing Assembler by SLR. 75K lines/minute, MASM compatible (except 80386, CodeView local symbols). Segmentation directives, procedure local labels, all symbols public, pseudo-ops. MAKE. MS \$ 159

AI Expert System Dev't

Arity Combination Package PC \$ 979
CxPERT V3.0 - shell for C MS \$ 349
Level 5 - formerly Insight 2 MS \$ 589
T.I.: PC Easy PC \$ 435
Personal Consultant Plus PC \$2589
Turbo Expert-Startup(400 rules) PC \$ 119
Corporate (4000 rules) PC \$ 329

AI-Languages

A.P.T. - Active Prolog Tutor PC \$ 49
ARITY Prolog - Interpreter PC \$ 229
PC Scheme LISP - by TI PC \$ 79
TransLISP - learn fast MS \$ 79
TransLISP Plus w/runtime MS \$ 99
TURBO PROLOG by Borland PC \$ 69
Turbo Prolog Toolbox PC \$ 69

C Language-Compilers

AZTEC C86 - Commercial PC \$ 499
C86 PLUS - by CI MS \$ 359
High C Optimizing Compiler PC Call
Instant-C - source, debug. MS \$ 349
Instant-C/16M PC Call
Lattice C - from Lattice MS \$ 259
Microsoft C 5.1 - Codeview MS \$ 289
Microsoft Quick C MS \$ 69
NDP C-386 by MicroWay MS \$ 529
Rex-C/86-standalone ROM MS \$ 695
Turbo C by Borland PC \$ 67
Watcom C6.0 - by Watcom Products MS \$ 259

C Language-Interpreters

C-terp by Gimpel - full K & R MS \$ 219
C Trainer - by Catalytix PC \$ 89
Interactive C by IMPACC Assoc. PC \$ 189
Run/C Professional MS \$ 145
Run C PC \$ 79
Turbo C-terp PC \$ 119

C Libraries-Files

BTree Source, no royalties MS \$ 69
CBTREE-Source, no royalties MS \$ 109
ctree by Faircom-no royalties MS \$ 309
ctree w/ rtree MS \$ 519
dtree MS Call
rtree - report generation PC \$ 239
dB2C Files MS \$ 259
db_Query MS Call
dbQuery MS Call
dbVISTA - Object only MS Call

FEATURES

MacFortran/020 by Absoft. Optimized FORTRAN-77 with VAX, 8X extensions, 68881 support, arrays memory or disk, complex math. Source debug, toolbox, C interfaces.

MAC 512, SE, II \$449

Foxbase PLUS by Fox Software. DBMS conforms to standard MAC interface. Up to 9 definable output windows, definable buttons, import/export hypercard stacks. MAC, Developer Kit. \$339

"Keeping Professional Programmers Current with Developer's Technology."

Programmer's Update lives up to its motto with a viewpoint unique to this industry. We use our experience and contacts to bring you articles and intriguing software trends and technical issues, interviews with authors and innovators, news about products, surveys, insightful commentary and predictions, valuable resource listings. You can get a FREE sample copy today by calling our toll-free number. Mention "DD688." A personal subscription is just \$25 a year.

Our Services:

- International Sales Desk
- Compare Products
- Help find a Publisher
- Evaluation Literature FREE
- Programmer's Update
- Dealers Inquiry
- Newsletter
- Rush Order
- Over 700 products
- National Accounts Center

C Libraries-General

Blackstar C Function Library PC \$ 99
C Tools Plus - V5.0 PC \$ 99
C Utilities by Essential PC \$ 119
Entelekon C Function Library PC \$ 119
Greenleaf C Sampler PC \$ 69
Greenleaf Functions PC \$ 129
LIGHT TOOLS by Blaise PC \$ 69
Turbo C Tools by Blaise PC \$ 99

C-Screens, Windows, Graphics

C Display Manager PC \$ 109
C Power Windows by Entelekon PC \$ 129
C Worthy Interface Library PC \$ 249
Curses by Aspen Scientific PC \$ 109
dBASE Graphics for C PC \$ 69
ESSENTIAL GRAPHICS-fast PC \$ 185
GraphiC - new color version PC \$ 279
Greenleaf Data Windows PC \$ 155
w/source PC \$ 259
Panel/TC - for Turbo C PC \$ 95
PC Forms-by Golden Software MS \$ 109
QuickWindow/C PC \$ 75
ScreenStar - with source PC \$ 169
Terminal Mapping System PC \$ 279
TurboWINDOW/C-for Turbo C PC \$ 75
View Manager - by Blaise PC \$ 199
Vitamin C - source, menus PC \$ 159
VC Screen - screen paint PC \$ 79
Windows for C - fast PC Call
Windows for Data - validation PC Call
ZView - screen generator MS \$ 139

DataBase & File Management

Advanced Revelation PC \$ 779
CQL PC \$ 329
DataFlex by Data Access PC \$ 595
DataFlex multiuser PC \$1049
Magic PC PC \$ 169
Paradox V2.0 List: \$725 PC \$ 499

DBASE Language

Clipper compiler PC \$ 399
dBASE III Plus PC \$ 399
dBASE III LANPack PC \$ 649
dBx "dBASE To C" translator MS \$ 469
Source to Library MS \$ 339
DBXL Interpreter PC \$ 99
FoxBASE+ - V2.0 MS \$ 259
Quicksilver Diamond PC \$ 369

DBASE Support

dAnalyst PC \$ 219
dBASE Tools for C PC \$ 65
dBRIEF with BRIEF-Auto-Indent, views structures PC Call

RECENT DISCOVERY

VM/386 Multitasker by IGC. Unlimited DOS programs in virtual 8086 mode. EGA in background, auto switch to background, separate RAM configurations, prioritize, determine number of cycles per task. PC \$229

DBASE Support cont.

dBC III by Lattice MS \$169
dBC III Plus PC \$509
dbug III MS \$179
Dialect UI PC \$ 45
dFLOW - flow charts MS \$125
Documentor - dFlow superset MS \$229
Genifer by Bytel-code generator MS \$249
Networker Plus - by WordTech MS \$229
QuickCode III Plus MS \$189
R&R Report Writer MS \$139
Seek-It - Query-by-example PC \$ 79
Silver Comm Library MS \$139
Tom Rettig's Library PC \$ 79
UI Programmer-user interfaces PC \$249

Debuggers

Breakout - by Essential PC \$ 89
CODESMITH - visual PC \$ 99
C SPRITE - data structures PC \$119
DSD87 PC \$ 79
Periscope II PC \$139
Periscope II-X PC \$105
Periscope III-10 MHZ Version PC \$959
Pfix-86 Plus - by Phoenix PC \$209
SoftProbe II - embedded systems PC \$695

Editors for Programming

BRIEF Programmer's Editor PC Call
de - EMACS-style PC \$ 65
EMACS by UniPress Source: \$895 \$265
Epsilon - like EMACS PC \$149
KEDIT - like XEDIT PC \$ 99
ME Macro Editor - Source PC \$159
MKS VI MS \$ 65
PC/EDT - macros PC \$229
Personal REXX - V1.6 PC \$ 99
SPF/PC - Version 2.0 PC \$179
Vedit PLUS MS \$129

Fortran & Supporting

ACS Time Series MS \$399
Fortran Addenda PC \$139
I/O Pro - includes No Limit PC \$229
MACFortran by Microsoft MAC \$229
MS Fortran - 4.0, full '77 MS \$289
NDP Fortran-386 by MicroWay MS \$529
PC-Fortran Tools - xref, pprint PC \$165
RM/Fortran MS \$399
Scientific Subroutines - Matrix MS \$129

Multilanguage Support

BTRIEVE ISAM MS \$185
BTRIEVE/N - multiuser MS \$455
GSS Graphics Dev't Toolkit PC \$375
Halo Development Package MS \$389
HALO '88 PC \$209
Help/Control - on line help PC \$ 99
Hoops 3D Graphics Library PC \$549
Informix 4GL-application builder PC Call
Informix SQL - ANSI standard PC Call
Instant Programmer's Help MS \$ 79

Note: Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add \$3/per normal item. All prices subject to change without notice.

We support MSDOS (not just compatibles), PC DOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees, and every product for Microcomputer Programming.

Real-time Programmers:

Build multitasking into your applications, add communications and interfaces easily, write blazing code with a C-like assembler.

Choose a real-time multitasking O/S, work with a classic real-time toolbox.

Call a Tech Rep TODAY!

	List	Normal	SPECIAL
C Sharp Real-time Toolkit - includes graphics	\$495	\$479	\$399
DESQview API Tools - multitasking, interfaces	\$550	\$449	\$419
Greenleaf Comm Library - complete, for C	\$185	\$129	\$105
QNX - real-time multitasking O/S	\$650	\$629	\$599
RisC - high level ASM C-like	\$ 80	\$ 75	\$ 59
Timeslicer - multitasking	\$295	\$265	\$239

Order before June 30, 1988 and mention "DD688" for these special prices above.

Multilanguage Support cont.

NET-TOOLS - NET-BIOS	PC \$129
Norton Guides	PC \$ 75
Opt Tech Sort - sort, merge	MS \$ 99
Pfinish - by Phoenix	MS \$209
Report Option - for Xtrieve	MS \$109
Screen Sculptor	PC \$ 89
Seidl Make (SMK) - large apps	MS \$ 89
Synergy - create user interfaces	MS \$329
XQL - SQL for Btrieve	MS \$649
Xtrieve - organize database	MS \$179
ZAP Communications - VT 100	PC \$ 89

OS Support

DOS Merge 286	PC \$ 139
Microsoft Windows Development Kit	PC \$ 69
MKS AWK	PC \$ 315
MKS Toolkit - Unix vi, awk	MS \$ 65
Norton Utilities Advanced	PC \$ 115
System V/AT Combination	MS \$ 99
Xenix System V 386	PC \$ 479
Xtree - Professional	PC \$1145
	MS \$ 109

Translator

dB2C Toolkit	MS \$ 249
For_C by Cobalt blue	MS \$ 659
Promula.Fortran 66 to C obj.	MS \$ 429
SofTRAN - Translation Lang.	PC \$ 349
TP2C - by BISS	PC \$ 199
Turbo-to-C-Tools by TGL	PC \$ 479

Other Languages

ACTOR	PC \$ 409
Ada Dev's Toolkit-Vol.1 & 2	PC \$ 889
Alslys Ada w/ 4 M RAM	PC \$2995
APL*PLUS/PC	PC \$ 439
CCS Mumps - Multiuser	PC \$ 359
Microsoft MASM	MS \$ 105
Modula-2 - V3.0 Dev. System	PC \$ 199
Smalltalk/V	MS \$ 85
Smalltalk/V 286	PC \$ 159
SNOBOL4+ - great for strings	MS \$ 80

Other Products

ASMLIB - 170+ routines	PC \$ 125
Back-It by Gazelle	MS \$ 119
Baler	PC \$ 459

RECENT DISCOVERY

OS/2 Programmer's Toolkit by Microsoft. Programmer's Reference, Learning Guide, on-line help, OS/2 utilities. Two hours modem support. OS/2 \$279

Other Products cont.

CO/SESSION - remote access	PC \$229
Dan Bricklin's Demo II	PC \$169
Disk Technician-smart upkeep	PC \$ 89
Easy Flow V5.0	PC \$125
Fast Back Plus	PC \$149
Flash-Up	PC \$ 69
Link & Locate - Intel tools	MS \$309
Mace Utilities	MS \$ 85
Plink 86 Plus - overlays	MS \$275
PC-Metric - analyze complexity	MS \$ 89
PVCS Corporate by Polytron	PC \$339
PVCS Personal	PC \$135
R-DOC/X	MS \$135
Sapiens Make	MS \$155
Show Partner F/X	PC \$328
Seidl Version Manager	MS \$269
Source Print - V3.0	PC \$ 75
TLIB	PC \$ 89
Tree Diagrammer	PC \$ 65
Visible Computer: 8088	PC \$ 65
WKS Library by Tenon	PC \$ 79

Xenix/Unix

Cobol - by Microsoft	\$639
Fortran or Pascal-by Microsoft	\$429
FoxBASE+	\$649
RM/Cobol	\$959
Xenix Complete System 286	\$979

Translation with Structure

BAS_C not only translates QuickBASIC 3.0 or BASICA to C, it also transforms your program into indented, structured C code.

Quick
BASIC
↓
C

- Large memory model BASIC runtime library included
- Scopes variables into local or global
- Unlimited nesting
- Accepts QuickBASIC 4.0 DO-LOOP, FUNCTION
- Options like BASIC image comments, brute conversion
- Microsoft, Turbo, Lattice, Aztec, or Quick C

Economy BAS_C translates programs of up to 1,000 statements. Commercial BAS_C handles programs of unlimited size and includes runtime library C source for all memory models and compile options.

Economy BAS_C	\$199	PS Price: \$179
Commercial BAS_C	\$375	PS Price: \$349

Gotoless Conversion

Box 835910, Richardson, TX 75083 214-404-1404

CIRCLE NO. 198 ON READER SERVICE CARD

SCREEN GENERATOR - WINDOW MANAGER

POPSCREEN

FOR PROFESSIONAL LOOKING PROGRAMS

Powerful But Simple Display Editor

Makes easy work of creating complex & colorful displays, pop-up menus, windows. Features easy boxes, outlines, block moves, superimpose, complete color & character graphics control, boilerplating, on line help; & much more.

Displays Integrate With Your Compiler's Output

PopScreen outputs your displays to library modules for your compiler. Displays link into your program at link time. No frustrating resident loaders, no loading screens from files on disk. Your displays are part of your program!

Instant Screen Access

Your displays pop to the screen instantaneously - in 1/25th of a second! Open and close windows with a simple procedure call. Your displays are fast, professional, and optimally compressed for storage inside your program.

PopScreen 3.0 Supports:

C: Microsoft, Lattice, Turbo, IBM
8088: will output assembly code
PASCAL: IBM, Microsoft
TURBO PASCAL: inline code
QUICKBASIC: library modules
DBASE III+ loadable .bin files

PopScreen

only \$89.00
PS Price: \$79

BaySoft

Box 5662-P, Albany, Cal. 94706
415-527-3300

60 DAY SATISFACTION GUARANTEE

CIRCLE NO. 199 ON READER SERVICE CARD

Call for a catalog, literature, advice, and service you can trust.

800-421-8006

HOURS: 8:30 AM. - 8:00 P.M. E.S.T.
5-D Pond Park Road, Hingham, MA 02043
Mass: 800-442-8070 or 617-740-2510 4/88

THE PROGRAMMER'S SHOP
Your complete source for software services and answers

ware implementations favor the separate FP stack provided by the hardware. Software implementations favor the efficiency of having everything on the normal (data) stack. Judging by the discussions at the last ANS Forth meeting, the separate FP stack wins hands down. Still, you can have your cake and eat it, too. Imagine a definition of *F+* such as the following in a software FP system:

```
: F+ [FPOP] (add numbers here)
      [FPUSH] ;
```

In a Standard program, *[FPOP]* would pop an FP number from the FP stack and push it on the data stack. *[FPUSH]* would have the reverse action. Nonstandard programs that required maximum speed would simply redefine *[FPOP]* and *[FPUSH]* before compiling the same FP extension:

```
: [FPOP] ; IMMEDIATE
: [FPUSH] ; IMMEDIATE
```

The existence of a separate FP stack implies the existence of FP stack operators, and most FP packages supply *FDUP*, *FDROP*, *FSWAP*, *FOVER*, and *FROT*. In some packages, *F@* and *F!* move FP numbers to and from *FVARIABLES*, and *FCONSTANT*s such as *PI* aid readability. FP numbers can be compared with *F<* and *F>*. *F=* may be provided, but comparing floating-point numbers for equality is a questionable practice and should be avoided. It is meaningful, however, to compare

an FP number to 0 with *F0=* or *F0<*. *FABS*, *FNEGATE*, *FMAX*, and *FMIN* are usually also available.

How to input an FP (real) number varies from Forth to Forth. Most Forths install some kind of automatic conversion to FP when the FP extension is compiled. In polyFORTH, numbers containing a period and followed by at least one nonblank character are converted to FP—for example, *2.0E* would be a “real” 2 whereas *2.* would be a double-precision 2.

In MasterFORTH, UR/FORTH, and MacFORTH, all derivatives of the FVG Standard, real numbers must contain an uppercase *E*—*EE*, or *2.E*, or *2.0E*, or *2E0*, and so on.

You must be in *DECIMAL* or strange things happen. To print a real number, use *F.*, but first set the number of places to the right of the decimal with *PLACES*:

```
4 PLACES
3.14159E F.
```

prints 3.1416.

The polyFORTH *F.* is an exception to this rule and takes the number of places from the stack. If you are a polyFORTH user and would like compatibility with the FVG Standard, redefine *F.* in this way:

```
VARIABLE Places
: PLACES ( n ) Places ! ;
```

```
: F. ( r ) Places @ F. ;
```

Let's stop here for now. You can see that without much effort the floating-point packages of the various Forth vendors could be brought into alignment, and this is one of

the things the ANS Forth committee is trying to do.

Now let's see what's involved in writing a floating-point package, in case you would like to experiment with it. First, you need a proper representation. The most efficient representation for Forth seems to be to use a normalized, double-precision signed mantissa and a single-precision signed exponent with the exponent on top. This gives you about nine digits of precision and an unheard of dynamic range. The high bit of the mantissa is the sign bit. The remaining bits form a normalized mantissa, unsigned, with the second highest bit set to 1. The binary point is to the left of this bit. This gives the mantissa 31 bits of precision. The number 1 would be:

```
HEX 0000 4000 1 DECIMAL
```

or “one-half times two to the one power.”

To multiply and divide these essentially double-precision numbers, you're going to need some extended math operators, shown in Table 1, below.

Writing the various stack-manipulation and memory-access words is also straightforward: *FDUP*, *FDROP*, *FSWAP*, *FOVER*, *FROT*, *F@*, *F!*, *F.*, *FCONSTANT*, *FVARIABLE*, and *FLITERAL*.

Rather than support a fancy system of infinities and “not-a-numbers,” the only special case I will handle is 0. A zero number will have a zero mantissa, which can be easily tested with the expression *OVER 0=*, as follows:

```
: QNORM ( q - t exp)
\ normalize q to bit 30;
\ leave adjustment as exp.
2DUP OR 2OVER OR OR
IF 1 ( count) >R
  BEGIN DUP 0< NOT
    WHILE Q2* R> 1- >R REPEAT
  2>R SWAP DROP 2R> TU2/ R>
  THEN ;

: ROUND ( t - ud exp )
\ assumes hi bit is zero.
32768 0 0 T+ ROT DROP
DUP 0< DUP IF >R DU2/ R> THEN ;

: F* ( r r2 - r3)
f0- IF FSWAP THEN
FOVER F0- IF FDROP EXIT THEN
( exp2 ) >R ROT ( exp ) >R UNPACK >R
2SWAP UNPACK >R DUM*
QNORM ( t exp ) >R
ROUND ( d exp ) R> + ROT ROT
2R> XOR PACK ROT 2R> + + 1+ ;
```

Example 1: Normalizing and rounding the quad-precision result

Forth Word

Meaning

D>SHIFT (d u - d2)	Double-shift d] [u bits to the right arithmetically
DU2/ (d - d2)	Unsigned double-precision right shift
T+ (tA tB - tC)	Add two triple numbers
T2* (t - t2)	Shift triple-precision number left once
TU2/ (t - ut)	Signed triple-precision number right shift
Q2* (q - q')	Shift quad-precision number left once
DUM* (ud1 ud2 - uqproduct)	Multiply unsigned double-numbers to yield unsigned result
DUM/ (uq ud - udquotient)	Divide unsigned quad uq by unsigned dividend ud
DUM*/ (ud ud2 ud3 - ud4)	(ud * ud2)/ud3 with quad-precision intermediate

Table 1: Extended math operators


```
: f0 = ( r - r f ) OVER 0 = ;
: F0 = ( r - f ) ROT 2DROP 0 = ;
```

Changing the sign of the mantissa is easy, but you must not change the sign of 0:

```
: FNEGATE ( r - r2)
\ 2's complement of R.
OVER IF >R [ HEX ] 8000
[ DECIMAL ] XOR
R> THEN ;
```

```
: FABS ( r - r2)
\ absolute value of R.
>R [ HEX ] 7FFF [ DECIMAL ] AND
R> ;
```

Packing and unpacking the sign bit from the mantissa is also easy:

```
: UNPACK ( d - ud sign)
DUP FABS ;

: PACK ( ud sign - d)
[ HEX ] 8000 [ DECIMAL ] AND OR ;
```

Probably the simplest function to implement is F^* because exponents add and mantissas multiply. You do, however, need some way to normal-

ize and round the quad-precision result, as shown in Example 1, page 94.

The other arithmetic primitives— $F+$, $F-$, and $F/$ —follow a similar pat-

```
: F. OVER >R FABS DUP 0>
IF 0 0 ROT 0
DO Q2* LOOP Q2* 999999999. DMIN
2SWAP DU2/
ELSE NEGATE D>SHIFT 0 0 2SWAP THEN
500000000. 1073741824. DUM*/
<# # # # # # # # # #
[ ASCII . ] LITERAL HOLD 2DROP
#S R> 0< SIGN #> TYPE SPACE ;

: FLOAT ( d - r)
2DUP D0= IF 0 EXIT THEN
SWAP OVER DABS 1 ( count) >R
BEGIN DUP 0< NOT
WHILE D2* R> 1- >R REPEAT
DU2/ ROT PACK R> 31 + ;

: >F ( d - fn)
\ converts most recent double number
\ to mixed fraction.
\ Used like 3.14159 >F
DPL @ 0< ABORT" Needs dec point"
FLOAT
DPL @ 2DUP
IF 1 0 ROT 0
DO 10 0 DUM* 2DROP LOOP
FLOAT F/
THEN ;

3.14159 FLOAT F. prints 3.141589999
3.14159 FLOAT FCONSTANT PI
PI PI F* F. prints 9.869587719
```

Example 2: Input and output primitives

tern. For a simple four-function package, you lack only the primitives for input and output, shown in Example 2, this page. $>F$ converts a double to a real number, and $F.$ prints it.

This simple version of $F.$ clips the real number to only a part of its dynamic range, limiting it to 999999999. Numbers less than 0.000000001 are printed as 0. A more sophisticated version would use logarithms to change the number from a power of 2 to a power of 10 before printing.

Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call 415-366-3600, ext 221. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

Vote for your favorite feature/article.
Circle Reader Service No. 6.

CQL QUERY SYSTEM

Both for \$395.00 ANSI X3.135 COMPATIBLE

Add SQL compatible ad-hoc query capability to your new and existing applications

Layered system includes CQL Interpreter, embedded CQL Library, Portable Windowing System, Screen I/O System, and Report and Form Generation Systems.

Complete C Source code included.

Hardware Independent

Interfaces provided for IBM/screen memory, IBM/BIOS, MS-DOS generic (ANSI.SYS), and Xenix (table driven multi-terminal interface adaptable to other multi-user systems).

Compiler Independent

Tested with Microsoft V4.0, Lattice V3.1, Lattice V2.15, Aztek (Manx), Xenix System V Version 1.2.

File System Independent

Interfaces provided for C-tree (trademark of Faircom) and BTRIEVE (trademark of SoftCraft Inc.).

Complete I/O Control

Data types include 8-bit binary, 16-bit binary, 16-bit unsigned binary, 32-bit signed binary, Monetary, 32-bit floating point, 64-bit floating point, 32-bit date, 32-bit time.

Machine Independent Software Corporation
1415 Northgate Square #21B
Reston, Virginia 22090 (703) 435-0413

Portable Application
Support System

PASCAL → C

Turbo Translator: TPTC

- * Translate your Turbo Pascal programs (v 3.x) to Turbo C.
- * Designed specifically for Turbo C.
- * Compares favorably against Microsoft's Quick C Translator!
- * Affordable Price! Our price is not a misprint!
- * Saves you hundreds of hours
- * A few hours of your time is worth more than the price of the TPTC!

Introductory Offer: \$49.00

- Ship. & Handling: U.S. & Canada (\$5) Foreign (\$20)
- * Checks (except traveler's checks) drawn on foreign banks: add \$30.00
- * Purchase order processing fee: \$10.00

CHEN & ASSOCIATES, INC.

4884 Constitution Avenue, Suite 1-E
Baton Rouge, Louisiana, 70808

(504) 928-5765 (Inquiries)

800-448-CHEN (orders only)

Telex: 650-3418200

FAX: (504) 928-9371

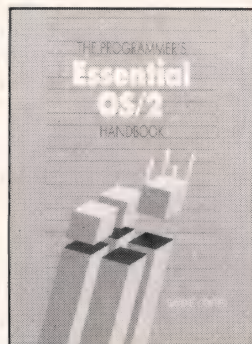
Trademarks: Turbo Pascal & Turbo C (Borland), Quick C (Microsoft)

CIRCLE NO. 146 ON READER SERVICE CARD

CIRCLE NO. 102 ON READER SERVICE CARD

POWER UP!

THE PROGRAMMER'S ESSENTIAL OS/2 HANDBOOK



by David E. Cortesi

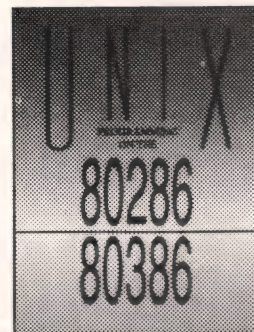
For writers of OS/2 programs, *THE PROGRAMMER'S ESSENTIAL OS/2 HANDBOOK* will give you the OS/2 technical information that will enable you to write efficient, reliable application in C, Pascal or assembler. Yet, it is organized in a way that won't let you get lost in its intricacies. Multiple indices and a web of cross referencing provide easy access to all OS/2 topic areas. There's even detailed technical information that is not included in the official OS/2 documentation. Equal support for Pascal and C programmers provided.

Inside, you'll find an overview of OS/2 architecture and vocabulary, including references to where the book handles each topic in-depth; a look at the 80286 and a description of how the CPU processes data in real and protect mode; an overview of linking, multiprogramming, file access and device drivers; an in-depth discussion of important OS/2 topics, including dynamic linking, message facility, the screen group, inputs, outputs, the queue, the semaphore, and more.

THE PROGRAMMER'S ESSENTIAL OS/2 HANDBOOK is written in straight, unambiguous language. This 700 page book is a resource no programmer developing in the OS/2 environment can afford to be without.

Book & Disk (5-1/4" & 3" OS/2)	Item #89-5	\$39.95
Book only	Item #82-8	\$24.95

UNIX PROGRAMMING ON 80286/80386



by Alan Deikman

UNIX PROGRAMMING ON 80286/80386 provides experienced system programmers with an overview of time-saving Unix features and an in-depth discussion of the relationship between Unix and DOS. Included are many helpful techniques specific to programming under the Unix environment on a PC.

Inside, you'll find complete coverage of the Unix program environment, file system shells and basic utilities; C programming under Unix; mass storage programs; 80286 and 80386 architecture; segment register programming; and Unix administration and documentation.

UNIX PROGRAMMING ON 80286/80386 contains many practical examples of the device drivers necessary to communicate with PC peripherals. It also includes useful information on how to set up device drivers for AT compatibles, such as cartridge tape drives and raster scan devices. Many examples of actual code are provided and are also available on disk.

Book & Disk (UNIX 5-1/4")	Item #91-7	\$39.95
Book only	Item #83-6	\$24.95

To Order: Return this form with your payment to: M&T Books, 501 Galveston Dr., Redwood City, CA 94063 or, **CALL TOLL-FREE 800-533-4372** Mon-Fri 8AM-5PM PST (IN CA, call 800-352-2002)

☐ **Yes!** Please send me the following:

Item#	Description	Disk Format	Price

Subtotal _____

CA residents add sales tax _____ %

Add \$2.95 per item for shipping _____

TOTAL _____

☐ Check enclosed. Make payable to M&T Publishing.

Charge my ☐ Visa ☐ MC ☐ AmEx

Card No. _____ Exp. Date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

MICRO/ SYSTEMS

JOURNAL
FOR THE
PC SYSTEMS
INTEGRATOR

SUBSCRIBE
NOW AND

SAVE
OVER
36%

OFF THE
NEWSSTAND
PRICE!



Yes! I want to subscribe to
MicroSystems
JOURNAL for the PC Systems Integrator

36%
Savings

and save over 36% off the cover price.

☐ 1 Year (12 issues) \$29.97 ☐ 2 Years (24 issues) \$56.97

Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed

☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

*Savings based on the full one-year newsstand rate of \$47.40. All foreign countries please add \$7 per year for surface mail; Canada & Mexico add \$17 per year for airmail; other countries add \$28 per year for airmail. All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank. Please allow 6-8 weeks for delivery.

A PUBLICATION OF M&T PUBLISHING, INC.

214S



Yes! I want to subscribe to
MicroSystems
JOURNAL for the PC Systems Integrator

36%
Savings

and save over 36% off the cover price.

☐ 1 Year (12 issues) \$29.97 ☐ 2 Years (24 issues) \$56.97

Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed

☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

*Savings based on the full one-year newsstand rate of \$47.40. All foreign countries please add \$7 per year for surface mail; Canada & Mexico add \$17 per year for airmail; other countries add \$28 per year for airmail. All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank. Please allow 6-8 weeks for delivery.

A PUBLICATION OF M&T PUBLISHING, INC.

214S



Yes! I want to subscribe to
MicroSystems
JOURNAL for the PC Systems Integrator

36%
Savings

and save over 36% off the cover price.

☐ 1 Year (12 issues) \$29.97 ☐ 2 Years (24 issues) \$56.97

Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment Enclosed

☐ Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

*Savings based on the full one-year newsstand rate of \$47.40. All foreign countries please add \$7 per year for surface mail; Canada & Mexico add \$17 per year for airmail; other countries add \$28 per year for airmail. All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank. Please allow 6-8 weeks for delivery.

A PUBLICATION OF M&T PUBLISHING, INC.

214S



No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

MicroSystems
JOURNAL for the PC Systems Integrator

Box 3713
Escondido, CA 92025-9843



No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

MicroSystems
JOURNAL for the PC Systems Integrator

Box 3713
Escondido, CA 92025-9843



No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

MicroSystems
JOURNAL for the PC Systems Integrator

Box 3713
Escondido, CA 92025-9843



SUDDENLY, MAGIC PC MAKES YOUR DBMS OBSOLETE

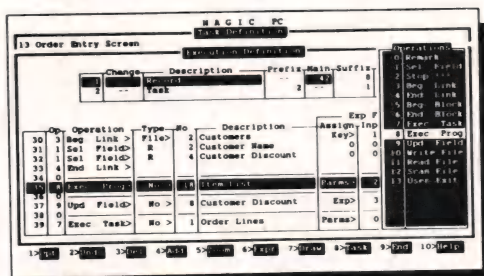
Attn
Btrieve
programmers:

You know how database applications are created — by hacking out line after line of time-consuming code. Most DBMS' and 4GL's give you some programming power. But when it comes to serious applications, they keep you bolted to your seat writing mountains of tedious code. And rewriting it all over again with every design change.

Imagine how much faster you'd be if you could replace the painful coding phase with an innovative visual technology which takes only a fraction of the time: Introducing Magic PC—the revolutionary Visual Database Language from Aker Corporation:

• High-Speed Programming:

With Magic PC's visual design language you quickly describe your programs in non-procedural Execution Tables. They contain compact programming operations which are executed by Magic PC's runtime engine. You fill-in the tables using a visual interface driven by windows and point-and-shoot menus. One table with 50 operations eliminates writing more than 500 traditional lines of code. Yet with Magic PC you don't sacrifice any power or flexibility.



With a powerful set of high-level non-procedural operations you program at only a fraction of the time.

• Maximum Power AND Simplicity:

With Magic PC, you can generate robust DBMS applications including screens, windows, menus, reports, forms, import/export, and much more! Plus, Magic PC has one of the friendliest user interfaces you've ever seen. Using Magic PC you can look-up and transfer data through a powerful Zoom Window system. Magic PC even lets you perform command-free queries.

• Btrieve Performance:

Magic PC incorporates Btrieve, the high-performance file manager from SoftCraft. This gives you exceptional access speed, extended data dictionary capabilities, and automatic file recovery!

• Virtually Maintenance-Free:

With Magic PC you can modify your application design "on the fly" without any manual maintenance. Magic PC automatically updates your programs and data files on-line! This also makes Magic PC an ideal tool for prototyping complete applications in hours instead of days.

• FREE Networking:

Magic PC comes complete with LAN features. Develop multi-user applications for your LAN with Magic's file and record-locking security levels.

• Stand-Along Runtime:

Distribute your applications and protect your design with Magic PC's low cost runtime engine.

• All For Only \$199:

Best of all, Magic PC is an unbeatable bargain. For a limited time, Magic PC's price has been reduced to only \$199! Yes, this is the same Magic PC that normally lists for \$695! And Magic PC eliminates the need for a separate DBMS, compiler, or application generator. It comes complete with all the tools you need to develop your own database applications instantly.



"Magic PC's data base engine delivers powerful applications in a fraction of the time... there is truly no competitive product!"

Victor Wright — PC Tech Journal

Order No 999 Order Date 99/99/99				Customer No 99999 Address AAAAA9999999999999999 9999999999999999999999			
Line	Item	Type	Description	Quantity	Unit Price	Total Price	
999	99999	A	AAAAAAAAAAAAAAAAAAAAA	9 999	999 999 99	999 999 99	
Item List							
No	Description	Type	Price				
999	AAAAAAAAAAAAAAAAAAAAA	A	999 999	99 999	Order Sum	999 999 99	
					Discount	999 999 99	
					Sub Total	999 999 99	
					Sales Tax	999 999 99	
					Order Total	999 999 99	
					In Stock	999 999	
					Total Orders	999 999	
					Avail to Sell	999 999	

19

25

31

37

43

49

55

61

67

73

79

85

91

97

103

Pop-up Zoom Windows run multiple programs per screen — with point-and-shoot data transfer between windows!

• \$199 — With A Money-Back Guarantee!

For a limited time, you can get Magic PC for only \$199. And even at this low price, Magic PC is risk-free. If you're not completely satisfied, simply return it within 30 days and we'll buy it back (less \$19.95 restocking fee). And if you'd like a preview, Magic PC's Tutorial Demo is available for just \$19.95.

But you'd better hurry — Magic PC's special \$199 price won't last long!

• Join The Magic PC Revolution

To unleash your DBMS design power, order your \$199 copy of Magic PC right now by calling toll-free or returning the coupon below.

ORDER NOW: CALL
(800) 345-MAGIC
In CA (714) 250-1718

MAGIC PC
The Visual Database Language
by **AKER**



Yes! I want to generate powerful applications much faster!

☐ Rush me my copy of Magic PC at the special promotional price of \$199 (add \$10 P&H, and tax in CA. International orders add \$30). I understand I can return Magic PC for a refund within 30 days, if I'm not completely satisfied.*

☐ Rush me a copy of Magic PC Tutorial Demo at \$19.95 (add \$5 P&H, and tax in CA. International orders add \$15).

Name _____
Company _____ Phone _____
Street Address (no POB) _____
City _____ State _____ Zip _____
☐ Check enclosed ☐ Charge to my: ☐ VISA ☐ MC ☐ AMEX ☐ DISC ☐ ELS
Account No. _____
Acct. Name _____ Exp. Date _____
Signature _____
Return to: **Aker Corp., 18007 Skypark Cir B2, Irvine, CA 92714**

System requirements: IBM PC, XT, AT, PS/2 or 100% compatible with 512K RAM, hard disk and DOS 2.0 or later. 54" format, not copy protected. Dealer pricing available. *Return policy valid in US only.

Aker, Magic PC, The Visual Database Language are trademarks of Aker Corporation. All other trademarks acknowledged. © Copyright 1987, Aker Corp.



DBMS PRO'S:
MAGIC PC
GIVES YOU MORE
POWER FASTER
BEYOND CODING
\$695
Now \$199.

Mixed Reviews on a Bunch of 4.0s



A few months back, Microsoft invited a bunch of us computer press types to an all-day shindig so they could tell us about the high-performance languages. They even gave us all sweatshirts so we could prove we were there (I use mine to sleep in, subliminal messages being what they are).

During one of those sessions, a presenter made the interesting comment. "We had things in the works for Pascal until somebody else came along and took the market away" he said. Given the main topics of the seminar, it seemed clear at the time, that Microsoft had abandoned the PC-based Pascal market to this unnamed "somebody else." It also seemed clear that they had chosen to dig their trenches around C and Basic. But now things are not so "clear," because the biggest name in little computer software has released its own Pascal 4.0.

The version level numbers are pure coincidence. These numbers will be short-lived anyway, since Borland is about to introduce Turbo Pascal 5.0. Borland's version of Turbo Pascal will have a debugger and thus go head-to-head with Microsoft's Pascal 4.0 Codeview connection, which in turn will trigger some newer and better thing from Microsoft, and so on, and so forth. You know how that goes.

Anyway, this month's column examines the new Microsoft Pascal. Some good news and bad news will be shared along with some compari-

by Kent Porter

sons to Turbo Pascal 4.0. Then we'll talk about another 4.0, about which there's only bad news: the new LIM EMS standard.

Looking at MS-Pascal 4.0

The first thing that hits you about Microsoft Pascal 4.0 is the quantity

of the documentation—a 3-inch pile of paper. This documentation consists of the standard *User's Guide* and *Reference Guide* (little has changed, if anything at all, from the previous version) plus a *Version 4.0 Update* of 68 pages. Additionally, the documentation includes two Codeview manuals (the basic book and an update), plus a guide to the new Microsoft Editor.

That should give you an idea of the major enhancements in Microsoft Pascal 4.0: Codeview and the editor. Less obvious, but equally significant, are an upgrade to the large memory model (the only model now available), full-scale support for Windows development, and an OS/2 option.

One of the great disappointments about these documents is Appendix C of the *Reference Guide*, which discusses the differences between Microsoft and "other Pascals." What other Pascals? UCSD, for crying out loud. Give me a break! Who are these guys trying to kid? If Microsoft wants back a piece of the Pascal action, they won't get it by pretending that Turbo doesn't exist.

Another great disappointment about the manuals is that Microsoft still hasn't put related information all in one place nor indexed the information in an intuitive manner. Seldom can you simply look up something in the Microsoft Pascal docs; it's a sort of scavenger hunt with hints cleverly concealed by the indexer. The new manuals are no improvement over the old. In fact, the new documentation compounds the problem because you must now

hunt through six manuals, compared to the earlier two. For example, it took me more than an hour to find out how to get a program into the symbolic format Codeview expects.

The compiler package runs on a DOS machine, but it can act as a cross-compiler for OS/2 development. An installation option lets you set up the system for OS/2 protected mode, OS/2 real mode plus DOS, or both. Link-time directives specify in which environment the executable runs. This is one area where Microsoft has a decided edge over Turbo, which hasn't yet heard of OS/2.

You can run this development system on a dual-floppy machine if you have masochistic tendencies. The manuals assume that you do, which I suppose is valid as a worst-case treatment. In that event, you'll have to swap diskettes in the A: drive three times to compile and link. A hard disk is so strongly recommended that it is almost mandatory.

Compilation is a two-step process with an optional third pass. The first two steps (PAS1 and PAS2) compile and optimize the source code into object form. The third step (called PAS3) produces an object code listing with numbered lines and a symbol table. The results are then linked to produce an .EXE file. The package comes with the latest segmented overlay linker (Version 5.01), plus an OS/2 linker called BIND.

Pass 1 of the compiler has a dozen command-line switches, each with a corresponding metacommand that can be embedded in the source file. In case of conflict between the command line and a metacommand, the metacommand wins. Most of these are pragmas for various check routines—subscript ranges, dereferencing of NIL pointers, stack overflow, and so forth. Some also pertain to the generation

"How to protect your software by letting people copy it"

By Dick Erett, President of Software Security



Inventor and entrepreneur, Dick Erett, explains his company's view on the

protection of intellectual property.

"A crucial point that even sophisticated software development companies and the trade press seem to be missing or ignoring is this:

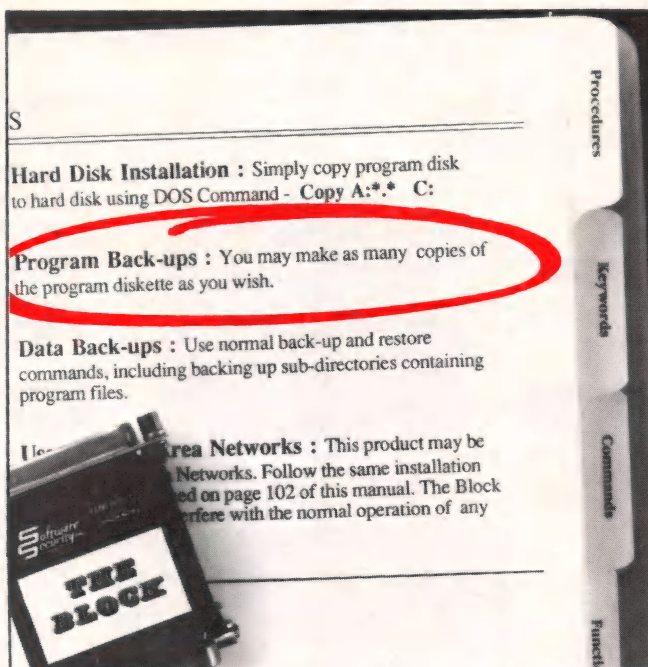
Software protection must be understood to be a distinctively different concept from that commonly referred to as copy protection.

Fundamentally, software protection involves devising a method that prevents unauthorized use of a program, without restricting a legitimate user from making any number of additional copies or preventing program operation via hard disk or LANs.

Logic dictates that magnetic media can no more protect itself from misuse than a padlock can lock itself.

Software protection must reside outside the actual storage media. The technique can then be made as tamper proof as deemed necessary. If one is clever enough, patent law can be brought to bear on the method.

Software protection is at a crossroads and the choices are clear. You can give product away to a segment



Soon all software installation procedures will be as straightforward as this. The only difference will be whether you include the option to steal your product or not.

of the market, or take a stand against the theft of your intellectual property.

"...giving your software away is fine..."

We strongly believe that giving your software away is fine, if you make the decision to do so. However, if the public's sense of ethics is determining company policy, then you are no longer in control.

We have patented a device that protects your software while allowing unlimited archival copies and uninhibited use of hard disks and LANs. The name of this product is The BLOCK™.

The BLOCK is the only patented method we know of to protect your investment. It answers all the complaints of reasonable people concerning software protection.

In reality, the only people who could object are those who would like the option of stealing your company's product.

"...eliminating the rationale for copy-busting..."

Since The BLOCK allows a user to make unlimited archival copies the rationale for copy-busting programs is eliminated.

The BLOCK is fully protected by federal patent law rather than the less effective copyright statutes. The law clearly prohibits the production of work-alike devices to replace The BLOCK.

The BLOCK attaches to any communications port of virtually any microcomputer. It comes with a unique customer product number programmed into the circuit.

The BLOCK is transparent to any device attached to the port. Once it is in place users are essentially unaware of its presence. The BLOCK may be daisy-chained to provide security for more than one software package.

Each software developer devises their own procedure for accessing The BLOCK to confirm a legitimate user. If it is not present, then the program can take appropriate action.

"...possibilities... limited only by your imagination..."

The elegance of The BLOCK lies in its simplicity. Once you understand the principle of The BLOCK, hundreds of possibilities will manifest themselves, limited only by your imagination.

Your efforts, investments and intellectual property belong to you, and you have an obligation to protect them. Let us help you safeguard what's rightfully yours. Call today for our brochure, or a demo unit."

Software Security inc.

870 High Ridge Road Stamford, Connecticut 06905
203 329 8870

OS/2 UNIX

■ WINDOWS ■ DATA ENTRY ■ MENUS ■ HELP MANAGEMENT ■ TEXT EDITING ■

New VCScreen 2.0

Vitamin C

PROFESSIONAL C LANGUAGE FUNCTION LIBRARY

- ☐ Multiple bullet proof overlapping windows
- ☐ Easy single field or full screen data entry
- ☐ Unlimited data validation
- ☐ Context sensitive help manager
- ☐ Menus like Lotus & Mac
- ☐ Programmable keyboard handler
- ☐ Text editor routines
- ☐ Printer output routines

- ☐ 30 day money back guarantee
- ☐ No royalties or runtime fees on applications
- ☐ Complete library source code included FREE

- ☐ FREE technical support
- ☐ FREE BBS at (214)418-0059
- ☐ Supports Microsoft 5, Quick C, Turbo C, Lattice and others
- ☐ Optional screen painter/generator

Better Applications In Less Time

Fast, flexible, versatile, reliable. Just some of the reasons why serious programmers use Vitamin C in their most important projects. They know using Vitamin C means lightning fast displays, a responsive user interface, professionally crafted C code, and a commitment to technical support.

High level functions provide maximum speed and productivity. Extended versions of these same routines add flexible control over specific details when necessary.

Versatile Design Keeps You In Control

Options and possibilities rather than limitations and frustrations mean you're always in control. Our versatile open ended design is full of hooks so you can intercept and plug-in your own control functions to easily customize or add features to most routines.

Easily create windows that pop-up, overlap, zoom, move, scroll, hide, show and resize. You'll choose options for titles, borders, colors, scroll bars, virtual size, and more. You can even access any window any time, even if it's hidden or invisible. That's flexibility.

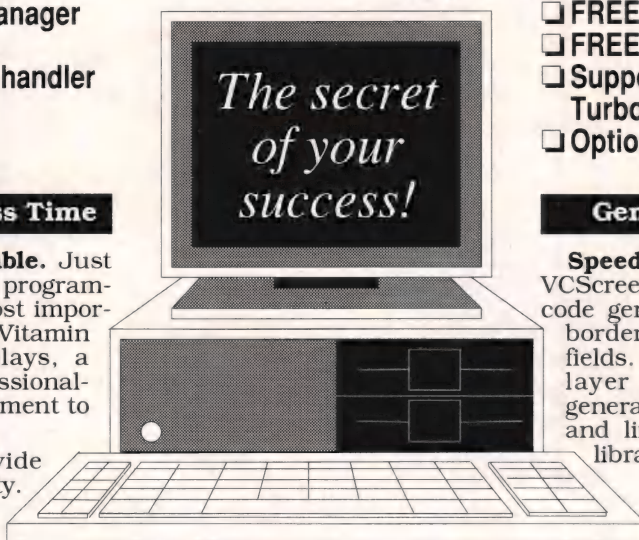
Sophisticated data entry forms become easy with features like unlimited validation, protected, invisible, and scrolling fields, full color control, single and multiple field input, selection sets, even right-to-left numeric input! And, with the context sensitive help system it's easy to provide field specific or other help messages.

Vitamin C's menus are the perfect framework for any application and feature advanced options such as check marks, unavailable items, blank items and separators.

The keyboard handler routines can redefine key assignments, translate keystrokes, even call a function.

Utility routines for time/date management, background processing, and sending windows to a printer.

Thorough documentation with tutorial and reference sections. Reference databases compatible with the Norton Guides Instant Access Program are also available.



Generate Code Inter

NEW VERSION!

Speed development even more with VCScreen, our interactive screen painter / code generator. Define windows, boxes, borders, headings, input and output fields. Copy, delete, change, move, even layer objects. Then let VCScreen generate C source code ready to compile and link with the Vitamin C function library.

New features allow creation of multiple windows, menu systems, global variable maintenance, user defined code

generation options, and more user configuration options!

Users And Reviewers Agree

"Picking the best value package is hard... If you're a source code fanatic like me, Vitamin C is preferable. If you need source code, make sure your wallet is wide open or get Vitamin C." Computer Language, June '87

"Only Vitamin C supports keyboard handlers and keyboard reassignment. Vitamin C provides the most options for menus." BYTE, October '87

"I trust our review of [Vitamin C] in Computer Language magazine was fair... it has become the screen manager package of choice at my firm." Michale Wilson, Wilsoft, Inc.

OS/2, UNIX and Xenix versions now available. Call for prices and details.

Vitamin C.....\$225⁰⁰

Includes source. Specify compiler when ordering.

VCScreen.....\$149⁰⁰

Requires Vitamin C library above.

Reference Database...\$50⁰⁰

Requires the Norton Guides program sold separately.

Requires IBM PC, XT, AT, PS/2 or compatible. Include UPS shipping: \$3 for ground, \$6 for 2nd day air, \$20 for overnight, \$30 if outside U.S. All funds must be in U.S. dollars drawn on a U.S. bank.

ORDER NOW!
(214)

416-6447

creative
PROGRAMMING

Box 112097 Carrollton, Tx 75011

STRUCTURED PROGRAMMING (continued from page 98)

of debugger information in the .EXE file. For a change, the user manual lists these metacommands on one page (in Table 5.2) and gives the defaults (all off). The other passes don't have switches, but the linker has nine. (All explained in Table 6.2.)

The 80X86 limits a segment size to 64K, which is the upward bound on any given code segment generated by the compiler/linker. The solution is to develop modular programs, compile them separately, and link them to form large applications. Each module then occupies its own code segment, thus allowing the program to transcend the 64K limit on individual segments. A routine in one segment can call a subprogram in another, so long as the external subroutine is declared PUBLIC in its owning module and EXTERN in the caller's; the result corresponds to a far proc in assembly language.

Microsoft Pascal's implementation of units is superior to that of Turbo Pascal 4.0. The Microsoft concept

more closely parallels Modula-2 (from which it was borrowed). In Microsoft Pascal a unit consists of two separate files, the Interface and the Implementation, and a mechanism exists for explicitly exporting identifiers. In Turbo, a unit is one file that consists of both parts, and everything in the Interface section is implicitly exported. The Microsoft approach more readily allows the implementation to be hidden from the caller, which is desirable in group projects and commercial packages.

The data segment of a Microsoft Pascal program is also limited to 64K. This segment contains all global variables, memory resident constants, the default heap, and (gulp!) the stack. Including the stack in the data segment is a bad design decision on the part of Microsoft, because a heavily recursive program with a lot of data can easily run out of stack space and have a nervous breakdown. Turbo does it better, with an entirely separate stack segment that can be up to 64K.

On the other hand, Microsoft of-

fers more flexibility with heap allocations. You can use the default (near) heap within the data segment, or the long heap located in all the uncommitted memory of the system. The only heap in Turbo is the long one, which can cause problems when one program spawns another. Microsoft provides a plethora of mechanisms for dealing with the two heaps.

At the aforementioned press briefing, Microsoft made a big deal out of its optimization, so I thought it might be instructive to test for it. One of the things Microsoft compilers purportedly do is to move invariant computations outside of loops. Here's a simplistic example:

```
FOR w := 1 to 100 DO BEGIN
  x := w + 10; { loop-variant }
  y := 12;     { invariant }
  z := (y * 13) DIV 7; { invariant }
END;
```

In other words, y and z need not be recalculated in each loop, since their values don't deviate from one iteration to the next. Optimization

AMX 86

KADAK's
engineers bring
years of practical real-time
experience to this mature

MULTI-TASKING SYSTEM

(version 2.0)

for the IBM® PC, PC/XT and PC/AT

- No royalties
- IBM PC DOS® support
- C language support
- Preemptive scheduler
- Time slicing available
- Source code of the C interface and device drivers is included
- Intertask message passing
- Dynamic operations:
 - task create/delete
 - task priorities
 - memory allocation
- Event Manager
- Semaphore Manager


AMX86™ operates on any 8086/88, 80186/88, 80286 system.

Demo package	\$25 US
Manual only	\$75 US
AMX86 system	\$2195 US

(shipping/handling extra)

Also available for 8080, Z80, 68000

KADAK Products Ltd.
206-1847 W. Broadway
Vancouver, B.C., Canada
V6J 1Y5
Telephone: (604) 734-2796
Telex: 04-55670



Amazing
COMPUTING
Your Original AMIGA® Monthly Resource



TRADITIONAL QUALITY & VALUE

Is Amazing Computing a little old fashioned?

At Amazing Computing™, we believe quality & value just make good sense. Each month, AC provides its readers with the finest techniques, reviews, and special features for the Amiga. Our staff has one objective: each issue is developed, shaped, and crafted into a publication our writers and readers want to read.

Amazing Computing enjoys a long line of firsts. AC was the first publication to document the Amiga's CLI, to show Amiga users how to improve video output, and to offer programs and inexpensive PD software while providing the first programming hints for Amiga Users. As the Amiga continues to grow, AC will continue to provide its readers with the most complete information available. With a past like Amazing Computing's, the future is our commitment to quality and innovation.

Save over \$23.40

As a subscriber to Amazing Computing, you are guaranteed to be informed and up to date on the rapidly growing Amiga market at a savings of over 49% of the newsstand price. A one year (12 issue) subscription is only \$24.00 (\$36 Canada & Mexico, \$44 overseas).

☒ **Yes!** Please start my 12 month subscription to the original Amiga monthly resource, Amazing Computing. I have enclosed \$24 for 12 issues (\$36 Canada & Mexico, \$44 Foreign Surface). All payments are US funds drawn on a US bank.

Name _____
Address _____
City _____ St _____ Zip _____

Send with a check or money order
(no billing or credit cards) to:
PIM Publications, Inc.
P.O. Box 869
Fall River, MA 02722

All rates are one year only. Foreign and US air mail rates available upon request. Please allow 6 to 8 weeks for delivery.

CIRCLE NO. 259 ON READER SERVICE CARD

CIRCLE NO. 187 ON READER SERVICE CARD

If it's all out warfare in today's software marketplace, you'd better have the best weapons.

Phar Lap 386 development tools. The best weapons.

Phar Lap 80386 development tools let you take full advantage of 386 protected mode architecture. You can break the 640K limit in the language of your choice; C, Fortran, Pascal, or Assembler.

For fast compact code, use 386|ASM, our full-featured 80386 assembler that's upwardly compatible with the MASM* 8086 assembler. Existing DOS and main-frame applications written in a high level language are easily ported by recompiling. And 386|LINK, our 32-bit native mode linker, puts it all together.

Debugging is made easy too. With our 386 symbolic debugger you can debug applications written in assembler or any high level language. Best of all, with Phar Lap's 386|DOS-Extender* you can run your native mode program on any 386-based PC running MS-DOS*. And you have full access to DOS system services through INT 21.

NO COMPATIBILITY PROBLEMS

Phar Lap's tools are compatible with the industry's leading systems: DESKPRO 386*, IBM Model 80*, accelerator boards such as Intel's Inboard* 386 and 386 clones. Not only will your new applications be compatible with the leading systems, they'll run alongside all other DOS applications.

NO ROYALTY PAYMENTS

Once your 386 application is complete, all you pay is a low one-time fee to license 386|DOS-Extender for redistribution. This allows you to embed 386|DOS-Extender in your application so your customers can run it on any 386-based PC. Just one payment and you unlock the entire DOS market. We don't believe in a software tax on every sale.

Don't wait for OS/3, get a jump on the competition today. Choose your weapons now.

- \$495 386|ASM/LINK — Package includes 386 assembler, linker, MINIBUG debugger and 386|DOS-Extender
- \$895 MetaWare 80386 High C* compiler
- \$895 MetaWare 80386 Professional Pascal* compiler
- \$595 MicroWay NDP Fortran-386* compiler
- \$195 386|DEBUG symbolic debugger

(617) 661-1510

PHAR LAP SOFTWARE, INC.
60 Aberdeen Avenue, Cambridge, MA 02138
"THE 80386 SOFTWARE EXPERTS"



Phar Lap and 386|DOS-Extender are trademarks of Phar Lap Software, Inc. MS-DOS and MASM are registered trademarks of Microsoft Corp. DESKPRO 386 is a trademark of Compaq Corp. Inboard 386 is a trademark of Intel Corp. NDP Fortran-386 is a trademark of MicroWay, Inc. High C and Professional Pascal are trademarks of MetaWare Incorporated. IBM Model 80 is a trademark of IBM Corp.

CIRCLE NO. 186 ON READER SERVICE CARD

STRUCTURED PROGRAMMING

(continued from page 101)

is supposed to place these calculations outside the loop.

Well, guess what? Microsoft Pascal doesn't optimize for this condition, regardless what Microsoft says. I wrote a little program with this kind of construct inside the loop, and then moved it outside. Here are the results:

With invariant calculations inside:
0.98 seconds

With invariant calculations outside:
0.44 seconds

Turbo doesn't claim to optimize for this, and their results for the same program versions were 1.10 and 0.49 seconds, respectively.

While on the topic of performance comparisons, I translated three of the standard Berkeley benchmarks from C into Pascal and ran them under both Microsoft and Turbo. The benchmarks are as follows:

sieve Tests array indexing and integer arithmetic
fib Tests recursion
acker Tests recursion and integer arithmetic

Execution times for the benchmarks on an 8-MHz AT clone with no-wait state memory were as follows:

	Turbo	Microsoft
<i>sieve</i>	40.97	31.36
<i>fib</i>	22.68	24.34
<i>acker</i>	12.36	12.58

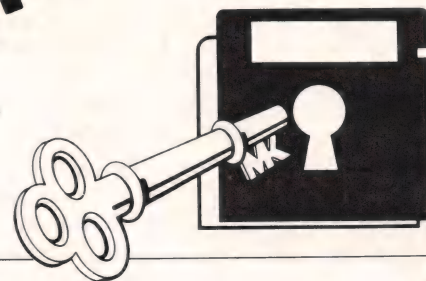
In other words, Microsoft does substantially better at *sieve*, and the other two benchmarks are somewhat of a wash. A comparison of code sizes reveals some startling differences:

	Turbo	Microsoft
<i>sieve</i>	2128	27,501
<i>fib</i>	2064	19,267
<i>acker</i>	2096	19,283

If the Microsoft compiler is so optimizing, why does it generate 10.5 times as much code on average to do the same job as the compiler from Borland?

MASTER*KEY

Unlocks Everything!



turn this
into this!

MASTER*KEY No Other Product Comes Close!

An EXPERT may not know the solution, but always knows where to find it.

MASTER*KEY HELPS ANYONE solve those confusing and frustrating software puzzles more rapidly and easily than any other software available, at any cost! It gives you know-how within hours that may otherwise take years of experience. Create a new program from an old one. DON'T REINVENT THE WHEEL!

MASTER*KEY - Smart!

MASTER*KEY is an intelligent self-documenting MS-DOS reverse assembler. Its sophisticated procedures swiftly race through massive and baffling object code files to effortlessly discover potential trouble spots.

MASTER*KEY - Educational!

YOU DON'T NEED TO KNOW ASSEMBLY LANGUAGE! MASTER*KEY will take any program from your IBM-compatible computer and return fully-documented, easily-understood assembly language source code (Microsoft MASM 5.0 compatible).

MASTER*KEY - Easy To Use!

MASTER*KEY works both automatically from the DOS command line or interactively from menus similar to Lotus Corporation's 1-2-3 or Symphony. No need to remember any new commands or continually refer to a manual. Use it immediately!

Minimum System Requirements:

256K + 8088/8086/80186/80286/80386 PC
MS-DOS or PC-DOS 2.0 +
One 360K DSD Floppy Drive (IBM PC Format)

MS-DOS is a trademark of Microsoft.
PC-DOS is a trademark of IBM.

C:\>DEBUG PROGRAM.COM

-D100 136

```
8848:0100 EB 18 49 6E 63 6F 72 72-65 63 74 20 44 4F 53 20 k.Incorrect DOS
8848:0110 76 65 72 73 69 6F 6E 0D-0A 24 50 B4 30 CD 21 86 version..SP40M!
8848:0120 E0 3D 36 01 72 05 3D 0A-02 76 09 BA 02 01 B4 09 '6.x...v...4.
8848:0130 CD 21 CD 20 58 EB 2F M!M Xk/
-Q
```

```
H00100: JMP      Short H0011A                ;00100 EB18    --
;-----
      DB      "Incorrect DOS version"        ;00102 49E636F727265
      DB      0Dh                            ;00117
      DB      0Ah                            ;00118
      DB      "a"                            ;00119 24
;-----
H0011A: PUSH     AX                          ;0011A 50    P
      MOV     AH,30h                        ;0011B B430    _0
      INT     21h                          ;0011D CD21    _!
      XCHG    AH,AL                        ;0011F 86E0    --
      CMP     AX,0136h                     ;00121 3D3601    -6
      JB      H0012B                       ;00124 7205    r_
      CMP     AX,020Ah                     ;00126 3D0A02    -v
      JBE     H00134                       ;00129 7609    v_
H0012B: MOV     DX,0102h                   ;0012B 8A0201    ---
      MOV     AH,09h                       ;0012E B409    ---
      INT     21h                          ;00130 CD21    _!
      INT     20h                          ;00132 CD20    _
;-----
;1-DOS_Ver_Number
;1-Display_String
;TERM_normally:20h
H00134: POP     AX                          ;00134 58    X
      JMP     Short H00166                 ;00135 EB2F    _/
;-----
```

MASTER*KEY XREF - PROGRAM.XRF

```
0102h      :      121    2F5    301    320
020Ah      :      126
03C8h      :      12B
1-Display_String      :      130    591    610
1-DOS_Ver_Number      :      11D
H00100      :      100
H0011A      :      100    11A
H0012B      :      124    12B
H00134      :      129    134
H00166      :      135
TERM_normally:20h     :      132
```

Page 1

NOTE: The cross-reference is by
memory location within
the program file!

NOTE: The output is totally
Microsoft MASM-compatible.

(not copy protected)

MASTER*KEY will guide you step by step to:

1. Help you learn assembly language, if you desire.
2. Discover how any program runs or why it doesn't.
3. Alter or remove unwanted object code from any program.
4. Incorporate routines from compiled programs into other assembly language, Basic, C, or Pascal programs.
5. Make software more compatible with your computer. Be certain a questionable program won't damage your system BEFORE you run it.
6. Modify software to operate with other versions of DOS.
7. Customize your COMMAND.COM or other executable program directly or by reassembling your altered MASTER*KEY source code.

Order Now!
Just \$79⁹⁵

Phone orders accepted on MC or VISA
\$82.45 (includes \$2.50 shipping)
\$87.65 in California (includes tax & shipping) C.O.D. orders add \$2.00

(714) 596-0070

Please send MASTER*KEY!

Send checks to:
Sharpe Systems Corporation
2320 E Street, Dept. 44, La Verne, CA 91750

Name _____
Address _____
City _____ State _____ Zip _____

Dealer/Distributor Inquiries Welcome

Sharpe Systems Corporation
2320 E Street, Dept. 44, La Verne, CA 91750 714-596-0070

MASTER*KEY should not be confused with any public domain or share ware software that may have a similar name or be a similar product.

CIRCLE NO. 214 ON READER SERVICE CARD

STRUCTURED PROGRAMMING (continued from page 102)

It happens that I make a living from nine to five as a software engineer in a big Unix house. I know from personal experience that you can optimize a compiler to recognize the standard benchmarks and to perform superlatively when it thinks its results have a chance of being published. This is optimization for the wrong reasons, inasmuch as benchmarks are synthetic programs that don't necessarily parallel real-world programs.

Suspecting that Borland might be guilty as charged, I wrote another program outside the realm of the standard benches. This one is SINES.PAS, which took 1000 sines on my AT clone (without a math coprocessor). Here are the results:

	Time	Size
Microsoft	5.12	19,413
Turbo	2.60	3,248

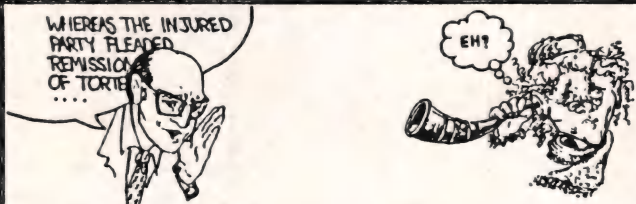
The upshot of this exercise is that Borland's minimally optimizing

compiler is either very smart indeed, or else it really does outperform Microsoft's by a 2-to-1 in timing and a 6-to-1 size ratio for a floating-point emulation application. So much for Microsoft's optimization.

Besides OS/2 support, the Microsoft compiler offers a couple of things that the Turbo compiler does not—Codeview and Windows. Codeview is a superlative debugger for .EXE files. It lets you set breakpoints and watchpoints, step through source lines and see where a program is going awry because of a runaway loop or whatever, monitor the stack, and so on. Maybe I'm old-fashioned; I still prefer to stick WRITELN statements into a misbehaving program to find out what's going on, but I make an exception where it comes to Codeview. It's great, and the ability to control debugging through three different means makes it the most intuitive debugger around. One of these days, we'll stack Codeview up against the soon-to-be announced Borland debugger and see how they compare. Meanwhile, Codeview wins the day.

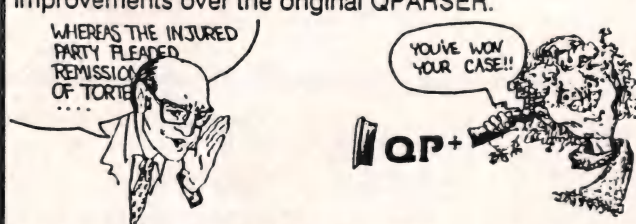
Microsoft claims that its C and Pascal 4.0 are the only languages that support Windows. That's not entirely true. For example, Actor is an object-oriented language for AI and Windows, which is Microsoft-approved. Although Actor is not a mainstream language, it does negate the assertion. Certainly the mainstream Turbo languages don't work with Windows; I struggled to write a Windows application in Turbo C and the program just wouldn't compile, despite help from Borland.

You could make the waggish observation that Microsoft's near-sole support is a commentary on the acceptance (or rather lack thereof) of Windows within the programming community. Nevertheless, the advent of OS/2 Presentation Manager—a glorified Windows—makes this unwieldy software the user interface of the future. For those who want to get a head start on that future, Microsoft Pascal 4.0 is an attractive alternative to the more arcane C language. Microsoft Pascal is the native language of Windows,



Now you can use QPARSER+ to develop compilers, interpreters, complex user-interfaces, language & file format translators (i.e. Pascal to C, Bit Map to Postscript), language debuggers like lint, etc.

Develop language translators in C and Pascal within the IBM PC/XT/AT or VAX/VMS environments. A new user manual, automated syntax tree construction and an advanced code generation language are just a few of the improvements over the original QPARSER.



Another translation by QPARSER+™
— FREE demo disk available
 Limited Time Offer
 Just \$300
QCAD Systems
 1164 Hyde Avenue, San Jose CA 95129 (408) 727-6884
 Outside Calif. call TOLL-FREE (800) 538-9787
 CIRCLE NO. 200 ON READER SERVICE CARD

EASY	FAST	PROFESSIONAL
DATA ENTRY WINDOWS MENUS HELP	GW Basic Quick Basic MS Cobol	Basic 86 Turbo C MS C Turbo Basic
Lattice C Quick C Instant C	If you are serious about programming PLEASE try HI-SCREEN XL!	
HI-SCREEN XL™ \$149 only Multilanguage support No Royalties 30-day risk free		
Mark Williams C RM-Fortran Microfocus Cobol	C Basic RM-Cobol Realia Cobol	dBXL Fox Base Turbo Pascal
Call now for demo and information: 1-800-338-2852 in CA: (415) 397-4666		
MS MASM MS Pascal	dBase II, III, and III+ MS Fortran	Turbo Prolog MS Basic Quicksilver
"You may like other screen management tools, but you will love HI-SCREEN XL."		

Softway, Inc., 500 Sutter St., Suite 222, San Francisco, CA 94102
 CIRCLE NO. 226 ON READER SERVICE CARD

which enforces the Pascal calling convention on C programmers who want to use its interface. Version 4.0 provides full access to the richness of the Windows environment by means of the `WINDOWS.INC` include file and the `$WINDOWS` metacommand.

Microsoft Pascal is purely a high-level language. Some of the intrinsic functions and procedures coat specific DOS calls with syntactic sugar. The language does lack general-purpose, low-level calls comparable to Turbo's `MsDos` and `Intr` procedures. Also, no direct access is provided to registers from Microsoft Pascal. If you want to do low-level stuff such as generating software interrupts, you have to write subprograms in assembly language and link them as externals with the Pascal program. This makes it necessary for serious developers to have an assembler (and guess who just happens to sell the best-known of them). It also complicates the development process. Microsoft hasn't been reluctant to extend the language in other areas, and they would have done their Pas-

cal users a great service by providing low-level calls and register access directly from the source level.

On the other hand, none of the major software manufacturers has its mixed language act together as well as Microsoft. You can link Pascal modules not only with assembly-language routines, but also with C and Fortran modules. Just as Microsoft C has an attribute for specifying the Pascal calling convention, Pascal also has a C-convention option. These modifiers govern the order in which parameters are physically passed to subprograms. Also, Pascal offers a `VARYING` attribute that allows Pascal programs to imitate C in passing a variable number of arguments to a subroutine.

Microsoft also has the advantage over Turbo in the areas of object portability and linkage. The Turbo linker is indistinguishable from the compiler and limited because it allows only the importing of assembly-language routines and the most rudimentary C routines. You can't export a Turbo Pascal object module because no such thing exists. Also,

Turbo doesn't support any overlays. The Microsoft linker is a separate program altogether, supporting mix-and-match of object modules coming from various languages and furnishing extensive support of overlays. Thus you can create programs in which each module is written in the language best suited for its task; Fortran for computations, assembler for low-level access and speed; Pascal for record processing; and C for control. The linker then joins them into a program in which little-used sections can operate as overlays. Pretty impressive stuff.

Also impressive is the new Microsoft editor, a welcome addition to the programmer's toolkit, which comes with Pascal 4.0. It's a windowing environment (though not a Windows application) that allows you to view and work on multiple source files, or different parts of the same file, in bordered text windows. The editor includes all the features you'd expect from an advanced program development tool. In effect, the editor creates a work environment much like Quick and Turbo, but

Now You Have A Choice

in Software Protection

CHOICE #1

The Secom Key

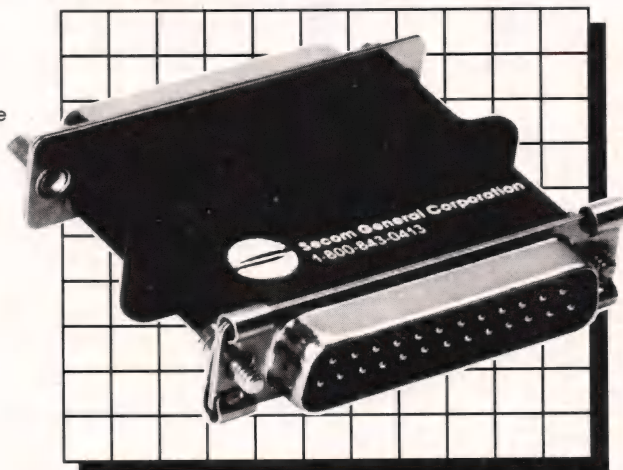
The Secom Key provides effective software protection while insuring customer satisfaction. It eliminates the problems normally associated with copy protection. The Secom Key is designed for software packages which are reproduced identically. Available in quantities, for as low as \$21.95.

Both the Secom Key AND The Memory Key:

- ☐ are completely transparent to end user
- ☐ will not interfere with peripheral operations
- ☐ don't occupy disk drives
- ☐ allow unlimited backup copies
- ☐ are easily installed
- ☐ are very small in size



Secom General Corporation
1829 E. Franklin Street #500
Chapel Hill, North Carolina 27514
(919) 942-8500



Secom offers alternatives!

If you would like a demonstration package or additional information, please write or call:

1-800-843-0413

CHOICE #2

The Memory Key

The Memory Key offers special flexibility. It comes with unique software which permits the use of its available read/write memory. Each byte of memory can be addressed individually or in groups for specific identification. Also, numerical information can be transferred between the Memory Key and your software and acted upon. The Memory Key comes with special error checking abilities providing 100% reliability. Example applications are:

- ☐ modular package control
- ☐ serialization
- ☐ software customization
- ☐ demo control
- ☐ auditable and easy software leasing
- ☐ any "counter" operation

The ease of use, cost effectiveness and functionality of the Memory Key allows for previously unavailable controls and applications.

CIRCLE NO. 210 ON READER SERVICE CARD

SLICK

OS/2
DOS
EDITOR

Before the SLICK editor was written, we evaluated many programmers' editors. All the editors had some features that were good. However none had it all: speed, ease of use, and features.

SLICK HAS IT ALL!!

- Edit first/last page without loading entire file
- Programmable file management
- Run programs concurrently (OS/2)
- Line, block and char marks
- Buffer and mark sorting
- Regular expression searching
- Window and file rings
- Command retrieval and completion
- Unlimited filesize
- Better word wrap
- Compiles 24,000 lines/minute
- Rexx-like macro language, can be used as OS/2 batch processor
- Add marked expressions
- Hex/octal/dec/floating pt. calculator
- SLICK compiler lints macros
- Automatic macro make
- Complete on-line help
- Syntax expansion/indenting
- OS/2 SLICK runs in DOS mode
- 30 day money-back guarantee

only \$99

MicroEdge Inc.

P.O. Box 2367
Fairfax, VA 22031

CALL (703) 378-4716

Runs on IBM PC/XT/AT
or compatible

IBM is a registered trademark of International Business Machines Corp.

CIRCLE NO. 160 ON READER SERVICE CARD

The Advanced
Programmer's Editor
That Doesn't Waste Your Time

For DOS, Microport
UNIX, SCO Xenix or

OS/2
Protected Mode

EPSILON

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
- Multiple windows, files
- Unlimited file size, line length
- 30 day money-back guarantee
- Great on-line help system
- Regular Expression search
- Supports large displays
- Not copy protected

Only \$195

Lugaru
Software Ltd.

5843 Forbes Avenue
Pittsburgh, PA 15217

Call
(412) 421-5911

for IBM PC/XT/AT's or compatibles

CIRCLE NO. 144 ON READER SERVICE CARD

STRUCTURED PROGRAMMING (continued from page 105)

more extensive because of windowing, a clipboard, and word-processing-like operations. It includes compile-link-and-go without leaving the editor.

In default mode the editor bears a visual and operational resemblance to Brief. The Microsoft editor is infinitely customizable with macros and provides the ability to assign the macros to keys. You can also write C routines and install them as editor functions. A control file called TOOLS.INI stores your configuration as ASCII text strings, which you can edit at will. The editor also comes with preconfigured files that make it emulate the Brief and Epsilon editors, the Quick environment, and Wordstar.

Will the new Pascal release win back the market for Microsoft? Probably not. It's harder to use than Turbo, doesn't offer as many extensions, takes much longer from source to executable, produces .EXE files that are unnecessarily large, and costs three times as much. For medium to light-heavyweight development, the only thing I can think of that makes Microsoft Pascal 4.0 preferable to Turbo 4.0 is Codeview.

On the other hand, Microsoft has now elevated Pascal into the same league with its venerated C and Fortran compilers for true heavyweight systems development. Especially strong are its support for Windows, OS/2, and mixed-language programming. These features are bound to win back some share of the Pascal market and to attract others who find C too esoteric for their tastes.

And Now For LIM 4.0

I had a terrific idea for this month's column. Four months after I ordered it on an emergency basis, Intel finally sent my LIM 4.0 upgrade kit. I thought I'd write a column about using the new EMS standard for interprocess communication. This made sense, since the focus of this month's issue is real-time programming, which implies multiple processes, that often need to communicate among themselves, right? Well, guess what. The upgrade kit didn't work.

My problems started with the first page of the slender leaflet. It said: trust me, all your problems are solved if you just run the INSTALL program. Uh-huh. Half a dozen panels into the program, the instructions tell you to please wait for the one second to several minutes required to find out how much memory you have. It was late, so I let the program survey all the world's memory while I showered. Then I rebooted, restarted INSTALL, and waited a while longer. Then I rebooted and re-restarted INSTALL, after that I went to bed. In the morning, it was still trying to quantify my paltry 1 Mbyte of EMS.

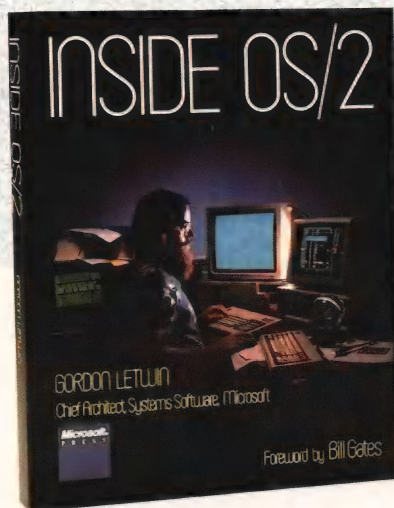
Since that failed, I tried to understand the gobbledygook on pages 3 through 10 of the upgrade instructions. The instructions purport to tell you how to manually install the upgrade. Trouble is, it's written in tongues. I speak nine languages fluently, and one of them is computerese, but my eyes glazed by page 4. In short, I couldn't get LIM 4.0 up and running, so I reviewed Microsoft Pascal 4.0 instead.

I griped about the LIM 4.0 problem among my computer buddies, and they all clucked and shook their heads. "A bunch of junk" was the consensus. Then Ron Copeland of DDJ relayed a rumor to the effect that LIM 4.0 is incompatible with the VGA, which I have on my AT. I discussed it over lunch with Tyler Sperry, who was still trying to recover from my original vitriolic attack on Intel. At his suggestion, I pulled my VGA and reinstalled the EGA from which I had upgraded shortly before.

Hey, suddenly it worked! Intel's INSTALL ran without a hitch. Whoopee, now I have more EMS capability and a lesser graphics board. Is the trade-off worth it? I don't think so. I shouldn't have to make that trade-off, nor should you.

It's no secret. If you look at the "Examining Room" column in this magazine, or in numerous other magazines, you will notice that I do a lot of product reviews. I don't like to be tough on little guys. They labor hard in their garages and there's much blood, sweat, and hope in what they produce. I have empathy with the little guys; I'm one of 'em.

The First Word on OS/2



"During the next 10 years, millions of programmers and users will utilize OS/2... The best way for them to understand the overall philosophy of the system will be to read this book." **Bill Gates**

INSIDE OS/2. Here—from Microsoft's Chief Architect of Systems Software—is a candid and exciting technical examination of OS/2. In unprecedented detail, Gordon Letwin explores the philosophy, key development issues, programming implications, and future of OS/2. And he provides the first in-depth look at each of OS/2's design elements—how they work alone and their roles in the system. INSIDE OS/2 is a valuable programmer-to-programmer discussion of the graphical user interface, multitasking, protection, encapsulation, inter-process communication, and more. You can't get a more inside view. **\$19.95.**

Microsoft® Press

Quality Computer Books

Available wherever books and software are sold.

Or call in your credit card order. **800-638-3030** (In MD 824-7300). Refer to ad DD38.

CIRCLE NO. 165 ON READER SERVICE CARD

Object-Oriented Engineering

If you don't pick the right environment, you may come up dry.

You've probably heard about object-oriented engineering. And the advantages it brings to software development.

You may even be ready to dive into it. We suggest you look before you leap.

Many companies today only offer one small bit of the support you need.

Not Stepstone. We give you more than just a compiler. We supply a whole object-oriented environment complete with powerful interpreter/debugger.

So you can shorten development schedules. Reduce maintenance costs. Build code which directly models your problem domain. And build truly re-usable software components.

Our products are based on the well known Objective-C® language and are designed for programmers who want the benefits of this new technology without giving up the advantages of C.

We even offer more than just a development environment.

We sell libraries of powerful, robust classes you can use directly in your applications. Like ICpak™ 201, a set of pretested Software-IC's for building custom graphical user interfaces.

Our products operate in a wide variety of workstation environments. As well as on the PC-AT™ and compatibles.

In addition, our support is second to none. We design, build, market and service our own products. Hence, we will do all that's necessary to keep you completely satisfied.

So before you jump into object-oriented engineering, talk to the company that's got the fullest range of products. Stepstone.

We won't let you down.

Stepstone™
The Leader in Object-Oriented Technology



The Stepstone Corporation 75 Glen Road Sandy Hook, CT 06482 203 426 1875 Telex 506127 FAX 203 270 0106

PC-AT is a trademark of International Business Machines Corporation. Objective-C and Software-IC are registered trademarks of The Stepstone Corporation. Stepstone and ICpak are trademarks of The Stepstone Corporation.

See us at Usenix Booth #109

CIRCLE NO. 231 ON READER SERVICE CARD

A lot of us are.

But Intel isn't a bunch of guys in a garage. They're a big outfit, and they ought to know better than to release this kind of flawed stuff. Months later, I might add.

I know how EMS works. I just wrote a chapter on it for my forthcoming book on advanced programming in Turbo C. EMS needs a 64K frame buffer in high memory, and it just so happens that VGA hogs a lot of memory exactly where EMS wants it.

So what we have here is a conflict of standards: VGA vs EMS 4.0. That's not my fault and I resent—on behalf of us all—having to make a choice between better memory management and better graphics. The way VGA works is hardly proprietary information. The two standards converge on one vendor, whose initials are IBM, which owns a substantial investment in Intel, who is the object of this diatribe.

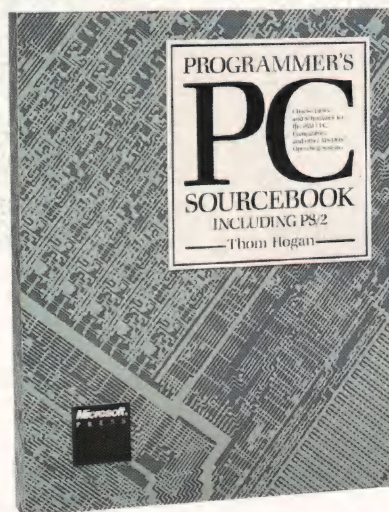
One could make a case that EMS was here first, and thus owns a legitimate stake in some unclaimed piece of high memory. On the other hand, VGA is hardware, which is by definition inflexible, whereas the EMS frame buffer is determined by a software device driver. The software ought to be smart enough to figure out how to coexist with VGA.

That it isn't is a discredit to all those Big Guys who allegedly have our best interests at heart. They screwed up, plain and simple, and it's you and I who have to pay the price for their inattention.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 9.**

Get Your Facts Fast



THE PROGRAMMER'S PC SOURCEBOOK.
The first place to turn for immediate, accurate information about your computer and its operating system. At last! Here is important factual information—previously published in scores of other sources—organized into one convenient reference. Designed to be your primary reference to information about IBM PCs and compatibles, PS/2s and MS-DOS, there are hundreds of charts and tables covering: ■ Keyboards, video adapters, and peripherals ■ DOS commands and utilities ■ DOS function calls and support tables ■ Chips, jumpers, switches, and registers ■ Other interrupts, mouse, and EMS support ■ DOS BIOS calls and support tables ■ Numeric conversions, character sets, and more. **\$24.95**

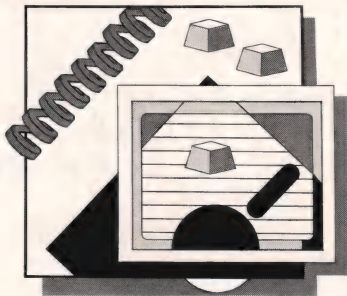
Microsoft® Press

Hardcore Computer Books

Available wherever books and software are sold.
Or call in your credit card order. **800-638-3030** (In MD 824-7300). Refer to ad DD68.
Book Code 86-96296.

CIRCLE NO. 167 ON READER SERVICE CARD

SD '88, Prolog Tools, and Transputers



There was an edge to some of the audience's questions. It didn't faze the panel of experts.

More than one member of the audience for the panel discussion on software development in the 1990s at Miller Freeman's Software Development '88 conference (SD '88) asked essentially the same question: "Today, the emphasis in object-oriented programming seems to be on software development as a creative activity. Can't we get a little more scientific, can't we move toward an engineering approach to object-oriented programming? More specifically, are there any rules or strategies you can point to that will help us decide what objects to define?"

The questioners got little satisfaction. The panelists' answers ran thus: Chuck Duff, author of the object-oriented programming language Actor: "A good plan is to study the physical system you are trying to model and create the classes of objects it has."

Dick Gabriel of Lucid, deeply involved in object-oriented Common Lisp: "That's a fundamental question for which there is no easy answer. I try things."

Bjarne Stroustrup, author of object-oriented C++: "It's a holy grail. There is no panacea."

Chuck Moore, author of the Forth programming language, which doesn't have to be object-oriented: "Programming is an art; we might hope it becomes a craft; it will never be a science."

by Michael Swaine

The panel members represented several programming paradigms besides the object-oriented one. Although there are object-oriented Forths and Lisps, those languages are surely paradigms unto themselves; and moderator Stan Kelly-Bootle was there to represent the

procedural paradigm single-handedly if he had to. (He's written extensively on Modula-2, has just finished writing a book on C, and was wearing an Ada T-shirt.) But with their combined knowledge of the object-oriented paradigm, one of them should have been able to field the insistent question if anyone could.

Maybe no one can. Maybe there is an invariant principle stating that, for any given paradigm and for any given state of the programming art (excuse me, discipline), there are aspects of the programming process that can be formalized and other aspects that can't. If so, surely a great deal of programmer effort has to be directed toward the latter. That's where only creativity will suffice. Maybe, as the panelists seemed to be saying, the decision regarding what objects to create when developing an object-oriented system is such an aspect.

Maybe. And maybe when you cross paradigm boundaries, the aspects shift. Then the first disorienting puzzle presented to you by a new paradigm would be to identify the problems for which only creativity will suffice.

On the Paradigms Beat at SD '88

Some of you may have attended SD '88. Ron and Jon and Allen and Tyler and I were there. SD '88 has grown in three years to become a truly important and informative conference for software developers. Of course, some of the sessions were less important and informative than others, and unless you knew the

speaker, it was a turkey shoot. The networking opportunities were good, though, as was the chance to see people whose work you've read. I had never seen Bjarne Stroustrup before.

I hope the conference sponsors can find ways to increase the conference's networking value next year. In addition to the sessions, the conference had exhibit space for companies, the main value of which was probably informal recruiting. Plans are for a greatly expanded exhibit program next year, and if that takes the direction of a sort of job fair it could be interesting.

This year there were tracks of lectures and workshops on artificial intelligence, database design, the C language, design methodologies, languages, and graphics. For the paradigmologist there was much to record. I attended many of the sessions I mention here, but since sessions ran concurrently I couldn't attend everything I was interested in. I'm summarizing some of the sessions from the proceedings.

While the speakers in the panel discussion I mentioned earlier weren't able to provide an engineering approach to deciding what object to develop in an object-oriented design, several sessions did deal with practical object-oriented programming issues.

The Oh-Oh Factor

Satish Thatte of TI talked about object-oriented database systems. OODB, Thatte argued, is a necessary step making smart front ends truly viable; conventional database architectures with AI front ends grafted on are handicapped by inflexibility. Citing the ten years it took relational database technology to be accepted commercially, he predicted that it will take OODB five to ten years to reach the market.

OODB represents a significant para-

digm shift for database developers. Object-oriented programming may be the paradigm shift challenging the largest number of programmers today.

Chuck Duff and Mark Solinski led a workshop on Actor development. Along with the developers of Smalltalk, Chuck has the distinction of having developed a commercially successful language strictly for object-oriented programming. Actor is a pure object-oriented language, down to the activation records on the stack (they're objects, too). I missed his talk, so I called him after the show and he gave me a little more insight into object-oriented programming in general and Actor specifically.

Chatting with Chuck

Chuck talked about multiple inheritance, the ability of an object to inherit from more than one parent, and about why he left it out of Actor and has no plans to add it. "At the implementation level," he said, "it turns simple tree traversal into arbitrary graph traversal. Somehow you have to linearize the graph. Smalltalk-80 did it by copying code, physically copying the methods." Duff called this a cop-out. Linearizing the graph is not impossible. "You can unfold the graph. But it adds code bulk," he said.

At the user level, Duff sees another kind of problem. He fears that users will view multiple inheritance as a panacea and misuse it. It is difficult to avoid conflicts, and the efforts necessary to do so may "make you wonder if you are really simplifying anything." He acknowledged that Lisp systems such as Flavors have implemented multiple inheritance, but he says he's seen some unreadable Flavors code result from that decision.

Having published last month some of Bjarne Stroustrup's views on what makes a language object-oriented, I asked Duff to talk about the defining elements of the object-oriented paradigm. "I think dynamic binding is fairly essential," he said. The Actor compiler works hard to convert dynamic bindings to static for efficiency, but the ability to use dynamic bindings supports what Duff calls "experimental program-

New! Hire a Pro for Your New Turbo 4.0

Turn on the power of Turbo PROFESSIONAL 4.0, a library of more than 300 state-of-the-art routines optimized for Turbo Pascal 4.0. You'll have professional quality programs finished faster and easier.

Turbo PROFESSIONAL 4.0 includes complete source code, comprehensive documentation and demo programs that are powerful and useful. The routines include:

- Pop-up resident routines
- BCD arithmetic
- Virtual windows and menus
- EMS and extended memory access
- Long strings, large arrays, macros, and much more.

Turbo PROFESSIONAL is only \$99.

Call toll free for credit card orders.

1-800-538-8157 extension 830

1-800-672-3470 extension 830 in CA

Satisfaction Guaranteed or your money back within 30 days.



Turbo Pascal 4.0 is required. Registered owners of Turbo Professional by Sunny Hill Software may upgrade for \$30. Include your serial number.

For other information call 408-438-8608, 9 AM to 5 PM PST. Shipping & taxes prepaid for US and Canadian customers, others please add \$6 per item.

TurboPower Software 3109 Scotts Valley Dr., Suite 122 Scotts Valley, CA 95066



CIRCLE NO. 240 ON READER SERVICE CARD

LAHEY SETS NEW FORTRAN STANDARDS

LAHEY PERSONAL FORTRAN 77 **\$95**

Low cost, Full 77 Standard, Debugger, Fast Compilation

F77L FORTRAN LANGUAGE SYSTEM **\$477**

For porting or developing, this is the critics' choice.

"Editor's Choice" *PC Magazine*

"...the most robust compiler tested." *Micro/Systems*

"...the most efficient and productive FORTRAN development tool for the DOS environment" *BYTE*

F77L-EM/16-bit **\$695** **F77L-EM/32-bit** **\$895**

Break through the DOS 640K barrier. The most powerful PC Fortran Language Systems for downloading or writing large programs.

PRODUCTIVITY TOOLS

Profiler, ULI Mathematical Functions Library, Overlay Linker, Toolkit, Utility Libraries, Windows, Memory Boards, 80386 HummingBoard.

IF YOU DEMAND THE VERY BEST, THEN YOU SHOULD BE USING LAHEY. CALL US TO DISCUSS YOUR PC FORTRAN NEEDS.

CALL FOR NEW FEATURES INCLUDING MATH COPROCESSOR EMULATION

30 DAY MONEY-BACK GUARANTEE

FOR INFORMATION OR TO ORDER:

1-800-548-4778

Lahey Computer Systems, Inc.

P.O. Box 6091, Incline Village, NV 89450

TEL: 702-831-2500 TLX: 9102401256

FAX: 702-831-8123

Lahey
Computer Systems Inc.

CIRCLE NO. 262 ON READER SERVICE CARD

ming." "Inheritance is necessary for reusing code. Ada packages are flat [do not support inheritance]," he added. Of course, the Ada people might disagree about the importance of inheritance, but he thought it fairly essential. "Encapsulation is widely accepted." Encapsulation, in conjunction with inheritance and dynamic binding, he said, is very powerful.

Returning to the question that had nagged the panel, he said, "We can be more scientific about it. We teach a course, and teach people to start with the physical system. The way its objects have evolved is probably a good way [to begin]." When the physical model is in need of redesign, he recommends doing systems analysis to find a better segmentation of the problem.

There will be more help for the object-oriented software engineer at this year's OOPSLA conference in San Diego in September, he said.

Back to SD '88: Another speaker, Rick Potter, discussed structured design for object-oriented programmers, pointing to a partial answer to the question that led off this column, but only a partial one. Yes, we can develop measures of goodness for objects and their interaction. No, that won't tell you what objects and classes to create.

What the Gods Would Destroy They First Submit to an IEEE Standards Committee

Object-oriented C was covered at SD '88 from several directions.

Lawrence Rosler predicted that the C programming language of the next generation will abandon at least one of the features that made the language popular: its compactness. C will get big, and will encompass alternative programming paradigms, certainly including object-oriented programming.

Bjarne Stroustrup gave an overview of C++, the proposed successor to C. He listed some of the features he left out of C++, including garbage collection, multiple inheritance (but AT&T has plans to add this), support for concurrency, ex-

ceptions, parameterized classes, and integration of the language with a programming environment. The benefits of these constraints, he explained, were compatibility, internal consistency, and efficiency. There were other workshops and lectures on C++ and Objective C.

Parallel Tracks

I found three talks that dealt with issues of parallelism.

Robert Ward had played around on the parallel machines at the Advanced Computing Research Facility at Argonne Labs and talked about programming large-scale parallel architectures such as Encore, Cray, and Hypercube. The focus of his talk was on shared-memory implementations, not communicating processes (although he claimed that the approaches are in some sense duals of one another, and that you can simulate one approach with the other). He argued the case for extending C to handle this sort of parallelism: the machines he was discussing all supported some form of Unix, which favors C, and C would make the programs more portable.

Since most parallel-processing work is in the experimental stage or is done for research projects where the developers don't see much need for portability, he had to justify this approach. Portability and maintainability are linked, he pointed out; also, the architectures are not stable. Moreover, developing a portable approach to parallel processing will facilitate the development of benchmarks for evaluating parallel architectures.

Finally, he answered the objection that machine specificity is necessary to get the performance benefits of parallelism. There are algorithmic benefits accruing from the use of a parallel approach, he said, but the benefits will be masked by fiddling with machine-dependent optimizations.

Mark Gluck and David Parker spoke cogently on neural networks, Gluck explaining why cognitive psychologists and neurophysiologists care about the stuff, and Parker sketching an algorithm.

Gluck argued that these models are more appropriately called parallel-associative networks since they

are in some ways not very neurallike at all. He sketched a brief history of associative net models, starting with the perceptron model of the early 1960s, which was unable to handle exclusive-OR; he told how Minsky and Papert shot down this model and everyone more or less abandoned the approach for a decade, and how it recently resurfaced when new algorithms were developed that implemented multilayer nets that do handle exclusive-OR.

Gluck has developed, with psychologist Gordon Bower, a model that accurately predicts human decision making in a medical prediction setting where the disease to be predicted is rare. Working with neurophysiologist Richard Thompson, he has applied a neural net model to sea slug neuron firing, with enlightening results.

Parker discussed an algorithm he developed for neural nets. He summarized the neural net approach succinctly. All learning is minimization, he said, generally minimization of error, and we have many good algorithms for minimization. The neural net approach is nothing but the parallelization of a minimization algorithm. His own algorithm is a parallel version of the steepest descent algorithm.

He pointed out that there was absolutely no performance advantage to the neural net approach over sequential minimization without parallel hardware. There may be design advantages, though.

Avram Tetewsky talked about tasking, the Ada facility for concurrent programming based on Tony Hoare's CSP language. He warned against the use of the simple task construct, in part because it limits your flexibility in passing data between tasks.

I Didn't Say That

Several speakers presented what one would normally think of as AI languages, and many of these speakers concentrated on non-AI uses of the languages. It looked as though the conference organizers had asked the speakers to demonstrate that AI tools could really be used for serious purposes.

Dick Gabriel talked about Lisp as a general development language and

as a systems language. He showed how to develop a sort of generalized spreadsheet using the Common Lisp Object System.

It was Gabriel, incidentally, who loudly ridiculed Sun's plan to re-write Unix in C++. Structured programming, he claimed, had threatened to stop cold the pace of advancement in software development in the mid-1970s, but, as it turned out, only retarded it for five years. C and Unix, he said, will stop us cold for 25 years. It was Dick Gabriel who said that, remember—not me. I'm just an innocent paradigmologist. Gabriel's at Lucid Inc., in Cambridge.

John Malpas presented Prolog in one workshop as an application language and in another in a software engineering context. He pointed out that the self-descriptive quality of the language makes it possible for a Prolog program to document itself to some extent.

To find out about SD '89, write to Miller Freeman, Seminar Dept., 500 Howard St., San Francisco CA 94105, or call 415-397-1881.

How Logical Is Prolog?

You know John Malpas's work: He did an article for us on Prolog. He and Dave Cortesi and I have made the most fuss over Prolog in these pages, and I wonder if I shouldn't feel a bit guilty about my part. There are a lot of people playing with Prolog today, and I choose that verb deliberately.

In a previous life, I was a consultant in research design and data analysis. It troubled me that many of my clients, all graduate students and faculty members, wanted to perform statistical analyses whose assumptions they did not understand. Now, as someone who has encouraged the widespread use of Prolog, I must take some of the guilt for the legions of Prolog programmers who don't know what resolution is.

To relieve my guilt, I'll tell you about a new book on Prolog that just came in. The book is *Prolog Programming in Depth* by Michael Covington, Donald Nute, and Andre Vellino (Glenview, Ill.: Scott, Foresman, 1988).

About half this book is spent de-

fining the language, which the authors do well and at a level an experienced software developer can appreciate. The discussion is strong on practical tips and bibliographic references, and on how features have been implemented in different compilers. There are also appendices on debugging and on features of Arity's and Borland's Prolog products.

The other half of the book presents artificial intelligence applications. There are no surprises in the selection of topics—search heuristics, expert systems, inference engines, natural language processing—or in the example programs the authors include.

Never Tell Me the Odds

The book is informed and informative. For example, the authors raise doubts about the confidence factors widely used in expert systems. Abundant research shows that people, expert or not, are poor at assessing conditional probabilities, and in fact at assigning numbers to just about anything. Confidence factors ought

Troff Support for Laser Printers!

EROFF™ now supports the HP LaserJet, Imagen, Laserwriter, and all POSTSCRIPT® printers!

CRT screen previewers for rough drafting are now included with **EROFF™**.

Full graphics previewers for SUN and X-Windows are also available.

Our enhanced troff allows inclusion of graphics directly into your documents. Available on MS-DOS and 36 different UNIX systems.

IMAGES



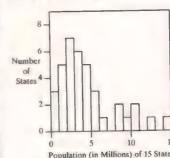
EQUATIONS

$$A'' = \lim \sum f(c_k) \Delta x$$

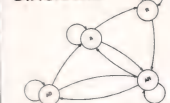
$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\partial^2 \phi}{\partial x \partial y}$$

$$\frac{b \pm \sqrt{b^2 - 4ac}}{2a}$$

GRAPHS/PLOTS



DIAGRAMS



TABLES

Food	Percent by Weight		
	Protein	Fat	Carbohydrate
Apples	4	5	13.0
Butter	18.4	5.2	—
Lima beans	7.5	8	22.0
Milk	3.3	4.0	5.0
Muesli	3.5	4	6.0
Rye bread	9.0	6	52.7

MICROSOFT, TURBO AND MIX POWER C PROGRAMMERS... C WINDOWS TOOLKIT PUTS YOU IN CHARGE OF VIDEO!

C Windows Toolkit is the only C programmer's windowing package that comes with a complete tutorial on monochrome, Hercules, CGA and EGA video. We don't just provide the functions, we also explain how to use them reliably.

And C Windows Toolkit comes with full, commented source code (would you trust a package that didn't?).

WINDOWING FUNCTIONS

- Create pop-up windows
- Create pull-down menus
- Create spreadsheet menus
- Create context-sensitive help screens
- Store windows for recall later
- Free memory used by windows
- Use 8 different types of exploding windows

SYSTEM SUPPORT

- Detect how many video adaptors are present
- Detect the types of video adaptor installed
- Switch between adaptors
- Detect ANSI.SYS
- Control the size and position of the cursor
- Detect the Enhanced Keyboard
- Disable the video signal
- Delay program execution to microsecond resolution

Over 80 functions that enhance your productivity.
Full source code included — No run-time royalties
Includes 200 page manual — 30-Day Money-Back Guarantee

EGA/VGA SUPPORT

- Use all 64 EGA colors
- Use EGA 43-line mode
- Use 2 fonts simultaneously
- Design custom fonts
- Smooth scroll the screen
- Smooth pan the screen

FAST SCREEN I/O

- Write to the screen lightning fast
- Write formatted output (like **printf()**)
- Get snow-free output on the CGA
- Scroll the screen
- Read characters off the screen

HERCULES SUPPORT

- Detect the presence of a Hercules Card
- Detect Ramfont support
- Load a Ramfont
- Switch between modes

Magna Carta
SOFTWARE

Requires: IBM PC, XT, AT, PS/2 or compatible that will run Microsoft C and/or Quick C
Supports Microsoft C 4.0/5.0/Quick C, Borland Turbo C 1.0/1.5, Mix Power C

From: **Magna Carta Software**
P.O. Box 475594
Gariand, TX 75047-5594
(214) 226-6909



Only \$99.95

(Texas residents add 8% sales tax)



Elan Computer Group, Inc.
410 Cambridge Ave., Suite A, Palo Alto, CA 94306
415.322.2450

CIRCLE NO. 121 ON READER SERVICE CARD

CIRCLE NO. 148 ON READER SERVICE CARD

to be examined with a skeptical eye, and these authors are appropriately skeptical.

They spend just one chapter laying the logical foundations of Prolog, but they deal with the implications of its logic throughout the book.

They do explain resolution and how Prolog uses resolution, producing proof trees via SLD resolution. They explain that SLD is sound (never letting you infer something that doesn't logically follow from other statements) and complete (find-

ing all possible inferences), and explain how Prolog's implementation of SLD resolution is sound but not complete, and they tell why it was implemented that way. They talk about the closed-world assumption and the way Prolog handles negation, and what these things imply.

Nevertheless, I wish the authors had said more.

They give practical advice on the use of the cut operator, but don't fully clarify the effects of cut, which some people fear can compromise the logic of a program. They should have said that the cut operator has no logical significance whatsoever:

Its use cannot change the logic of a Prolog program. Cut just prunes the proof tree, with a gain in efficiency but a loss in completeness.

What the authors refer to as "red" cuts are a special case. Here the programmer consciously writes code that is declaratively incorrect (logically incorrect), depending on his knowledge of the order of clause evaluation to keep the program from crashing. I wish the authors had come down harder on this kind of programming, which undermines any notion of Prolog as programming in logic, and ties the code to nonparallel implementations.

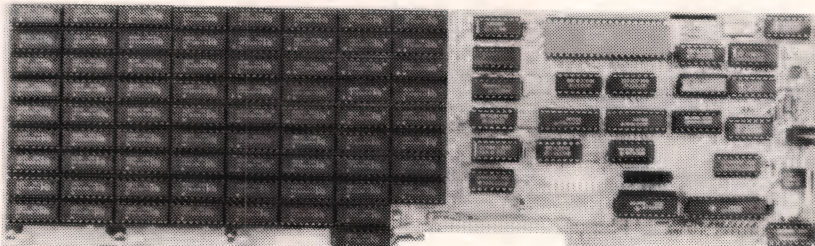
The principle of negation-as-failure and the closed-world assumption (CWA), both relevant to the logic of Prolog, are not equivalent. Frankly, I can't tell you how they differ, although I know that CWA is more powerful. But unless I missed it, the authors of this book don't clarify the point, and I think it is worth discussing in a book that examines Prolog programming in depth.

More Books

Despite these points, *Prolog Programming in Depth* is a good book. For the time being, though, the Prolog programmer who really wants to understand the logical structure of the language he or she is using may just need to read a book on the relevant aspects of logic. One on my shelf is *Foundations of Logic Programming* by J.W. Lloyd (New York: Springer-Verlag, 1987).

For the less committed, two good books that explain resolution briefly (in a chapter or appendix) are *Mathematical Theory of Computation* by Zohar Manna (New York: McGraw-Hill, 1974), an older book with 20 solid pages on resolution; and *Natural Language Understanding* by James Allen (Menlo Park, Calif.: Benjamin/Cummings, 1987). You would not buy either book just for their treatment of resolution, but both are books I thought you might have access to, and the Allen book is worth getting if you have any interest at all in natural language processing. I recommend them because I don't think it's reasonable to expect people who are merely experimenting with Prolog to buy and digest a book on mathematical logic. And most of the

GET A FLAME



The Blue Flame II is the latest in our line of very high-performance disk emulators for PC's, XT's, AT's, '386's, and all clones. It's extremely fast: 800Kbytes per second transfer rate, ten times faster than hard disks. Even faster than IBM's VDISK program! And big: Up to 8 megabytes per board, 32 megabytes per logical drive. Much bigger than extended or expanded memory. It doesn't waste any of your computer's memory address space for storage. And the Blue Flame II is reliable: With no moving parts, it can be accessed continuously for years with no failures. Don't try this at home with your hard disk!

Not just another RAMdisk, the Blue Flame II has an external AC-powered battery-backup option: Data isn't lost when the computer is turned off. And "Reset" isn't a dirty word anymore. Even during a blackout, the battery maintains data for 10 hours.

The Blue Flame II is available fully-populated, with 8 megabytes, for \$2095. 4 megabytes for \$1195. 2 megabytes for \$795. Battery Backup option costs \$135. Call us for information on our SemiDisk products for S-100, and Epson QX-10/QX-16.

If you want greater software speed, improved data security, increased hardware reliability, get a Flame. If you need the hottest disk performance possible, get a Flame. A Blue Flame II SemiDisk.

SemiDisk Systems, Inc.
P.O. Box GG
Beaverton, OR 97075
(503) 626-3104

WARNING:

This ad
contains strong
language.

Introducing Mic for OS/2

Microsoft Pascal 4.0

Compiler

```
File View Search Run Watch
0) i : 9
1) notprime : -10
14: writeln('
15: prime := 5;
16: repeat
17:   rprime := prime;
18:   sqrt := trunc(sqrt(rprime));
19:   i := 1;
20:   notprime := false;
21:   while (i < sqrt)
22:   begin
23:     i := i + 2;
24:     notprime := (rprime mod i = 0);
25:   end;
26:   if (not notprime)
27:     prime := rprime + 2;
28:   until (prime > 10000);
```

Microsoft BASIC 6.0

Compiler

```
File View Search Run Watch Op
Child$ : "dirsort\find " BAS"
FileNumber = 5 : 0.000000
' The child process does: D
Child$ = "dirsort\find " +
DIM Directory$(100) ' Stri
FileNumber = FREEFILE ' Mes
OPEN "PIPE:" + Child$ FOR I
WHILE NOT EOF(1) ' Loop un
LINE INPUT #FileNumber,
NumEntries = NumEntries
WEND
ChildDone: ' T
CLOSE FileNumber
FOR i=0 TO NumEntries
```

Microsoft C 5.1

Optimizing Compiler

```
File View Search Run Wa
0) i : 217
1) p : 23383:5936
125: int i
126:
127: set_cursor
128: p = screen;
129:
130: /* Draw top of box, +
131:
132: *p = 218;
133: p += 2;
134: for (i = 0; i
135: *p =
136: *p = 191;
137: p += 2;
138:
139: /* Draw side of box.
```

The people who co-developed the industry's most powerful personal computer operating system are now proud to announce programming languages to match.

Introducing Microsoft® Macro Assembler 5.1, C 5.1, Pascal 4.0, FORTRAN 4.1 and BASIC Compiler 6.0.

Five industrial-strength, stand-alone languages that combine the implementation flexibility you've enjoyed under MS-DOS® (which, of course, they still support) with the advanced capabilities you've anticipated from OS/2.

Capabilities such as the ability to develop

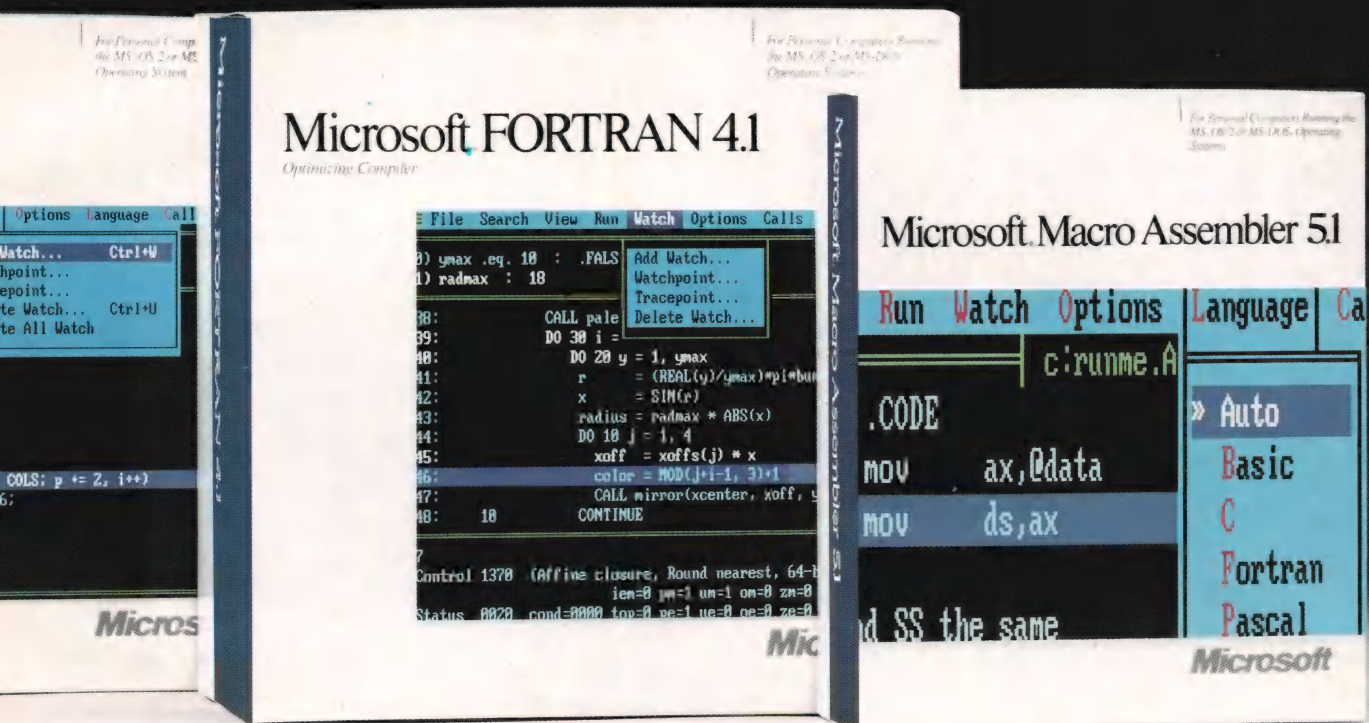
large, sophisticated applications which go beyond the 640K barrier, taking advantage of up to 16MB of RAM, and utilizing the potential of today's microprocessors.

Just like their MS-DOS predecessors, these five new languages are equipped with powerful, professional features you work with, not around:

Support of direct calls to the operating system, and inter-language calling for mixing multiple languages on the same project.

Access to OS/2 system calls and a full complement of utilities, including an incredibly fast incremental linker and the

Microsoft Languages systems.



first protected mode programmer's editor that works equally well in real mode.

Microsoft CodeView®, our popular, advanced debugger that lets you untangle program logic at the source code level, no matter what code you're using.

(It even lets you debug protected mode programs up to 128MB of virtual memory, and larger programs than ever before in real mode.)

As the perfect complement to our new languages, we're also offering the Microsoft OS/2 Programmer's Toolkit.

It contains a parameter-by-parameter

breakdown of all OS/2 system calls and samples to get you started.

All the tools you need for turning out larger, more powerful, more complex OS/2 applications.

(And, incidentally, all the tools we rely on for creating our own commercial software.)

For the name of your nearest Microsoft professional languages dealer, simply call 800-541-1261, Dept. F58

Ask him for some more information on our OS/2 family.

He'll show you some languages you can really swear by.

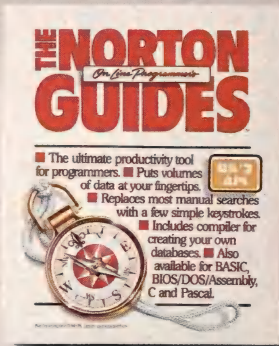
And now for the strongest word in the English language:

Free.

The Norton On-Line Programmer's Guide for OS/2 API is the first complete on-line manual for OS/2 programming.

Instead of thumbing through pages of documentation, it's all there at your fingertips with a few simple keystrokes.

Normally it costs \$150, but it's yours free when you acquire the Microsoft OS/2 Programmer's Toolkit and one of the high level languages listed opposite (an upgrade is fine).



Please send me my free copy of the Norton Guide for OS/2 API. I enclose a copy of my dated sales receipts and my registration cards*.

The high level language I have licensed is (please check):

- ☐ Microsoft C Optimizing Compiler 5.10.
- ☐ Microsoft FORTRAN Optimizing Compiler 4.10.
- ☐ Microsoft Macro Assembler 5.10.
- ☐ Microsoft Pascal Compiler 4.00.
- ☐ Microsoft BASIC Compiler 6.00.

Redeem to: Norton Guide for OS/2 API offer, Microsoft Corporation, 13221 SE 26th, Suite L, Bellevue, WA 98005.

Name: _____

Address: _____

City: _____ State: _____ Zip: _____

Daytime telephone: () _____

If you have any questions about this offer, call (800) 426-9400. In WA, (206) 882-8088.

F58

Microsoft®

*Registration cards are not required for upgrades. This offer is only valid in the 50 United States. It is not valid with any other offers, and is effective only for purchases from 4/1/88 through 6/30/88. The coupon must be redeemed by 7/31/88. Please allow 4-6 weeks for delivery.

purchasers of Prolog products today bought them precisely to experiment with the language.

Transputer Meditation

With any new paradigm, the first thing you want to do is experiment with it; see what its model problems look like and where it demands creative thinking, or creative rethinking about familiar problems.

Parallel processing is a radical paradigm shift, encompassing not just one class of new paradigms but a whole curriculum of them. The most radical of these can force you to rethink fundamentally how you approach familiar problems. I opened the topic of parallel processing here last month, talking about the INMOS transputer chip, and about occam, the language developed for programming transputers. I thought I had said about all I could until I actually got my hands on a transputer development system to play with.

But after I wrote that column, my Munich-based editor Jürgen Fey dropped by. He was in the country on a quick trip to SD '88 and other Silicon Valley attractions, and, as he usually does when he comes to California, he found time for a visit. We ate lasagna and drank California wine, swapped stories and the names of some good books, played with the dog and bounced on the trampoline, and finally, around midnight, we sat down and talked transputers.

Jürgen had been able to get his hands on a transputer development system to play with, and had been building a transputer board. He reminded me that, in the time since INMOS had designed the transputer chip to be used for parallel processing, a number of system development projects had been using transputers, proving the chip's practicality. Many of the projects were defense industry jobs, where details can be hard to come by and cost considerations differ from those in commercial markets. Nevertheless, such companies as Sun and Atari are now investing in transputers for commercial applications. Jürgen had

been bitten, too. He was eager to get back to Munich to finish his transputer board to show at the CEBIT show in March.

The Parts of TDS

I asked Jürgen if he was doing his development using TDS, the development system supplied by INMOS, and what he thought of it. He said he was, and that it was solid. TDS includes occam, a linker, an editor, a debugger, libraries, and a configurator.

It's the configurator, in part, that would allow Jürgen to develop parallel-processing software on a single-

transputer system if he wanted to. The configurator allows you to do your development work using one transputer, simulating a network of transputers in software, and then configure the program for a multiple-transputer system. You tell the configurator how many processors you have and where the links are, and that, I gather, is pretty much that.

Jürgen's initial system, though, contains two transputers, and that's because of the debugger. It's called a network debugger, and is particularly interesting, actually requiring a two-transputer system. The target program runs on one transputer,

Developing ROMs with Microsoft C™?

Complete it sooner with C_thru_ROM

C_thru_ROM—it works with Microsoft C to turn your PC into a complete ROM development workstation: complete debugging, complete locating, complete startup code, complete documentation, and completely self-contained. All to help you complete your project sooner.

COMPLETE DEBUGGING

Give hex dumps the dump. Use the remote debugger that's friendly and fast. C_thru_ROM allows you to debug on the target hardware directly from your PC! Debug at any level: source, assembly or mixed. Source-level debugging uses CodeView™ information. Windows are provided for viewing source code, machine registers, local and global variables, and commands. You also get complete execution control by tracing, on assembly or C-source line, breakpoints on expression and by line number.

COMPLETE LOCATION

The C_thru_ROM locator puts you in complete control of the location process. Locate code and data anywhere in 8086 memory and generate the output format you need—either Intel Hex, Intel Absolute OMF, binary image, or Tektronics Hex.

COMPLETE STARTUP CODE

Don't waste your valuable time writing startup code—it's already been done for you. C_thru_ROM includes startup code in source that's ready for ROMing. Everything's provided to take your 8086 from a cold start through setting the stack, heap, and segment registers, and calling main. It even has the hooks to handle stack checking, log critical errors, perform null pointer checks, etc.

COMPLETE DOCUMENTATION

C_thru_ROM's documentation won't leave you stranded. The package includes everything

from detailed program information to practical advice. Experienced ROM developers can go straight to the references they need, while learners of all levels can get assistance along the way from helpful suggestions and "how-to" instructions which are included with C_thru_ROM.

COMPLETELY SELF-CONTAINED

When you use C_thru_ROM you're using tools that were made to work with each other, and with Microsoft C, all on one PC. No more hopping from one machine to another, or trying to make hostile systems interact—every part of C_thru_ROM is designed for today's micro, not a rehash of old mainframe tools.

COMPLETE SATISFACTION GUARANTEED

Order your own C_thru_ROM development package and turn your PC into a complete ROM development workstation! If you're not completely satisfied, simply return it within 30 days for a full refund.

C_thru_ROM \$495

ORDER TODAY. Call Toll-Free

1-800-221-6630

Datalight

17505 - 68th Avenue N.E., Suite 304
Bothell, Washington 98011 USA
(206) 486-8086

Microsoft and CodeView are registered trademarks of the Microsoft Corporation.

Breakthrough in interface management. Generate C code from Dan Bricklin's Demo screens. Date fields. Full color support. Money fields. Fully programmable field behavior. Scrolling text within fields. Calculator style numeric input. User definable entry validation. Field marking. Orthogonal field movement. Specify fields by number or location. Source code included. Screen sizes limited only by memory. Interfaces with db_VISTA and other libraries. Text style numeric input. Input masking. List fields. Create spreadsheets. Includes Look & Feel screen designer. Integer fields. String formatting commands. Date and time validation functions. Generate C code with Look & Feel screen designer. Supports automatic vertical and horizontal scrolling. Clean screen fields per screen limited only by development. String fields. Easy to painting. Bind as much data as de data entry with commas. Ask a programming library. Hexadecimal or No fields. Float fields. Quick C. Speaker functions. Lattice. Create UNIX. Numeric validation. keystroke level. Customize screens 30 day money back guarantee. Gen assortment of editing commands. windows. Assign validation data to credentials. Pull down menus. Sup mode. All functions are kept in C style function reference. Pop-up functions. Numeric range checking. tive function names. Date and time Capture screens from existing as deep as desired. Easy to main checking. Date and time conver definition language based on C's ly definable borders. The current ally highlighted. Create reports.

Convert old programs to C. Borders with titles. Color map enables use of logical colors. Toll-free telephone support line. 24 hour bulletin board. Automatically detects type of monitor being used. ANSI driver included. Screen and field definitions. Uses device drivers for portability. View text in pop-up word entry fields. paging functions Customizable lect different cur Supports CGA, monochrome. cludes functions the display. Ask spreadsheet libra of keyboard Lined borders. menuing systems. scroll lights. Vid ver included. drivers can be cre map enables log colors. Borders lines. Fully inte system. Create as as needed. Create screens. Easy to ual. Professional port. Includes functions. Device drivers swappable at run-time. Context sensitive help system. Cross referenced help screens. Protected fields. Object-oriented design. Read in screen defini tions from disk files. Digitally mastered. Assign prompt strings to fields. UNIX. No run-time license. Numeric range checking. Unified field theory. Full printf % substitution

machine. Number of memory. Fast screen modify. Fast screen sired to fields. Numeric bout our linear pro fields. Long fields. Yes Read only fields. reports. XENIX and Validate data at the and menus at run time. eric data pointer. Rich Easy to learn. Pop-up fields. Corporate C ports EGA 43 line separate modules. Full prompt and message No royalties. Descrip- conversion routines. programs. Nest screens tain. Run time error sion functions. Screen printf. Time fields. Ful- field can be automati- Exploding borders.

C-scape 2.0

with



Look & Feel

"C-scape could send C programmers reeling." *PC Magazine*, 29 March 1988. Nominated for Technical Excellence 1987 by the editors of *PC Magazine*.

"I recommend the C-scape library." *Computer Language*, December 1987.

"C-scape is by far the best." *IEEE Computer*, February 1988.

Look & Feel™
Screen Designer

- WYSIWYG screen design tool
- Generates readable C code
- Create menus and data entry screens
- Define fields of any type
- Variables, prompts, and validation
- Line draw and erase
- Block, move, cut, paste, copy
- Horizontal and vertical scrolling
- Edit Dan Bricklin Demo slides
- Full color support
- Fast, easy, and fun to use
- Includes help
- Full-feature demo available

C-scape™ 2.0
Function Library and Toolkit

- Windows, windows, windows
- Menus, menus, menus
- Vast help system
- Create any type of field
- Data entry and validation
- Smart borders
- Extensive function library
- Swappable device drivers
- Easy to learn and use
- Easy to maintain and modify
- Unsurpassed flexibility
- Professional manual
- No royalties; no run-time license
- Source code included
- Demo package available

OAKLAND

675 Massachusetts Avenue, Cambridge, MA 02139-3309

800-233-3733 CALL
617-491-7311 NOW



PC/MS-DOS \$299 (price includes C-scape, Look & Feel, source and manual). UNIX/others call. 30-day review.

readable C code. Portable. Easily modifiable functions. No royalties. Source code included. Apollo and Data General. Professional support. Interface examples for data base management. Validation at keystroke level. Vast integrated and indexed context-sensitive help system. Save and restore regions of the display. Now supporting Quick C, Turbo C, Lattice, Microsoft, UNIX, XENIX, and others. And that's not all. Call for demo.

PROGRAMMING PARADIGMS (continued from page 119)

the debugger on the other. Jürgen says it's very powerful.

The folding editor is also interesting. It allows you to collapse detail, much as an outline processor does.

Jürgen then briefed me on the chip. There are three families of transputers now: the 16/32-bit T2xx, the 32-bit T4xx, and the 32-bit T8xx with an FPU. A transputer has four I/O ports called links, which facilitate the development of transputer networks. The occam language supports the links directly via what it calls channels.

Jürgen thinks the transputer is well-designed for parallel processing. In addition to the external parallelism it facilitates, there is a fair amount of parallelism inside the chip. Each of the four transputer links has DMA, and can perform memory accesses in parallel with each of the others and in parallel with the CPU, the FPU (if present), the ALU, and the integer unit.

Occam's Praisers?

Because of the transputer architecture and the nice match between the architecture and the occam language, many things you would like to be able to do with parallel processors are easy. Jürgen drew quick sketches showing how you would implement multiplexors and systolic arrays with transputers.

Some things, though, are not so easy. Occam is not a rich language, and C and Pascal programmers will find some of its limitations annoying. Its inability to do mixed-mode arithmetic, for example, is annoying. Its lack of operator precedence is alarming. And together these limitations can produce code that is full of parentheses and explicit type conversions.

Although occam is a high-level language, it permits some assemblylike optimizations. One of the most important lessons Jürgen learned was that indexes must be kept internal to the chip and large arrays external. Since the transputer may have 2K to 4K of internal RAM, this can become an issue.

But when you get beyond the optimization tricks, parallel processing

in any language can be a nightmare. Systolic arrays are a simple technique for implementing parallelism, but most of the parallel equivalents of sequential techniques are yet to be discovered. And Jürgen posed the question, how do you document a parallel-processing system for your boss/client? Nassi-Schneiderman diagrams won't work.

The Homebrew Computer Club vs. Japan, Inc.

One broad class of models that Jürgen thinks may prove fruitful is neural networks. Although the neural net model probably won't fit every parallel-processing problem, it is strictly parallel, and it works. Jürgen's next step will be to investigate neural net models. He thinks there are about 50 of them, and he'd like to get comfortable with at least 10 before he draws any conclusions about their usefulness for his goals.

Part of the appeal of parallel processing for me is that it forces you to jettison so much mental baggage. Jürgen, who also likes to travel light, likened the situation in parallel processing today to that of the Homebrew Computer Club in the 1970s, when hobbyists brought together their wire-wrapped boards and code and swapped ideas while hacking a trail to a new technology.

It's appealing to view parallel processing as a kind of hacker frontier, and that's not altogether wrong; but companies and governments with lots of money to spend have also been investigating parallel processing. Jürgen told of interviewing the head of Japan's ICOT, who talked about the Japanese commitment to research in parallel processing, and about their plan to develop an automatic parallelizer. The program would automatically convert any sequential algorithm to an efficient parallel form. The plan failed; the researchers had to settle for a simpler goal: developing a tool that would interact with a savvy programmer to help him parallelize the algorithm.

Jürgen said he took comfort in that failure.

Then Was Now and Now Is Then

Although I think it's clever, the above subhead doesn't really fit this clos-

If you see something here you like, compare it with what our competition has to offer.

Seidl Version Manager (SVM)

Local area network support provides conflict-free archive access to multiple users. • Archive database tracks source (and binary) file revisions. • Audit trail reporting provides information on project's development. • User definable and savable report formats enhance project review. • Revision merging and deleting allow flexibility in archive maintenance. • User IDs, privilege settings and passwords help resolve access conflicts, maintaining project integrity. • Optional text compression reduces storage requirements. • Menu driven shell simplifies access to complex, multi-featured tools. • Single-Site: \$299.95 + p&h • 5-site LAN: \$1000 (extendible)

Seidl Make Utility (SMK)

Uses structured language to define dependencies. • Supports rich command set with over 25 different statements. • Handles nested include files and library dependencies. • Runs faster than its competitors and handles larger projects. • Only \$99.95 + p&h • 5-site LAN w/ SMKgen: \$500

New! SMKgen

Automatically constructs dependency files that can be read by SMK. • Can also construct linker files and *filelist* files that integrate with SVM. • Performs consistency checking while building dependency files. • Only \$49.95 + p&h

"SVM is a full-featured system that has all the essential capabilities you are likely to need, and is further distinguished by several important features that none of its competitors yet have." - The C Users JournalTM, Feb. '88.

When you know the facts, you'll know we're the best.

**For more information call:
1-313-662-8086**

For a free copy of our competitive comparison report, call or write us on your company letterhead.

Visa/MC/COD Accepted

SEIDL COMPUTER ENGINEERING

3106 Hilltop Dr., Ann Arbor, MI 48103

CIRCLE NO. 211 ON READER SERVICE CARD

ing note. But "then was now and now is then" has been haunting me, and I knew that if I didn't use it soon somewhere in my writing, it would insert itself into my conversation in some even less relevant way, probably making me look like a fool. Looking like a fool in print is something every writer gets used to. I hereby place "then was now and now is then" in the public domain: feel free to use it as you dare. You may even find an appropriate use for it, and then you won't look like a

fool.

Five years ago, writing in *IEEE Spectrum*, Robert Kahn of DARPA gave this projection for computing in the 1990s:

Computer hardware: advanced packaging and interconnection techniques, ultra large-scale integration, parallel architectures, 3-D integrated circuit design, gallium arsenide and Josephson junction technology, optical components.

Computer software: concurrent languages, functional programming, symbolic processing (natural languages, vision, speech recognition, planning).

Computer performance: one giga-instruction per second to one tera-instruction per second.

Kahn was describing the fifth-generation computer technology, which the Japanese began planning for in 1979 and are pursuing with single-minded dedication today. I don't mean to hint that Kahn's predictions make him look like a fool. He may have missed on a couple of points, but some rough beast does seem to be forming out of the materials he inventoried, and the hour for some sort of fifth-generation computer technology's hour seems nearly at hand.

But whether it does slouch? Says Dick Gabriel: "Europe will be pouring six times as much government money into programming as the U.S. in the next decade. I expect the lead in software to move abroad."

Five years ago, it seemed plausible that the next generation of computer technology would be developed first in the United States. Today, based on funding and directness of effort, the most likely developer for fifth-generation computer systems is Japan, followed by a combined European effort, followed by the United States.

I guess it's a good thing we got all that practice reading their manuals when we bought their stereo systems.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 7**

THE MISSING LINKer

FIRMWARE DEVELOPMENT TOOLS for
MICROSOFT® C

LINK & LOCATE™++
SUPPORTS iAPX 86/87/186

**MICROSOFT®
C**

Downloads to
IN-Circuit Emulators with
INTEL™ OMF or
SoftProbe™ II Target Debugger

Includes...

- Start Up Files
- Linker
- Locator
- Library Support with Floating Point Operation
- COMPLETE Microsoft™ C Debugging Information

**ROMABLE
CODE**

**All This PLUS
Other Fine Products from...**

**Embedded
SYSTEMS & SOFTWARE, Inc.**



3303 Harbor Blvd. • C-11
Costa Mesa, CA 92626
714/241-8650

FREE
PRODUCT CATALOG
AND DIGEST...
WRITING
ROMABLE
CODE
USING
MICROSOFT C

NEW!

TOOLKITS FOR TURBO C & QUICK C from ZORTECH INC.

HOTKEY

A complete set of Terminate Stay Resident (TSR) functions that help you to write reliable 'pop-up' programs.

Now you can make your programs 'Sidekickable'. Two example programs are included, a 'pop-up Calculator' and a pop-up 'Critical Error Handler'.

The Hotkey toolkit handles all floating point functions in resident mode.

The 32 page manual includes an interesting discussion of the origin and history of undocumented MS-DOS function calls, together with a full explanation of the theory and practical use of TSR's.

Only \$49.95! (State Turbo C or Quick C version.)

COMMS

Do you need to incorporate serial communications into your applications? Yes! Then get this inexpensive but highly professional COMMS toolkit from Zortech Inc.

Look at the list of features: Xmodem, Kermit and ASCII file transfer, Hayes modem control, VT52, VT100 and ANSI terminal emulation, supports up to 8 serial ports, speeds up to 19.2k baud rate and higher.

Two demonstration programs are included, MINICOM and MAXICOM (like Procomm) together with the 120 page manual and full source code FREE!

Only \$49.95! (State Turbo C or Quick C version.)

GAMES

Have you ever wondered how to write a chess program? Now we reveal the secret algorithms and techniques of the masters with this dynamic Games toolkit.

The package comes complete with the full source code to three ready to play games of strategy - Chess, Backgammon and Wari (an ancient African game).

A comprehensive 150 page manual is provided giving an in depth look at the history, structure and program design of such 'Strategy Games'.

Only \$49.95!

(State Turbo C or Quick C version.)

SUPertext

This is not simply an 'Editor' toolkit, but a full-blown, 'WordStar' compatible wordprocessor with the full source code.

As well as all the normal editing functions, you will also find 'dot' commands and full printer control. The SuperText toolkit handles files of any size and allows full on-screen configuration.

Do you need to incorporate a wordprocessor into your application? Yes! Then get the SuperText toolkit complete with full source code and 150 page manual now!

Only \$49.95! (State Turbo C or Quick C version.)

PROSCREEN

Generate high quality data entry screens with the Pro-Screen - Screen Designer and Code Generator.

You can draw the data entry screen, define the input fields, define the input criteria, set screen colors and attributes, draw single or double lines, make boxes - press a few buttons and 'hey presto' Pro-Screen generates the C source code for your application!

Professional applications programmers will find this versatile utility and it's associated functions invaluable.

Comes complete with a substantial 78 page manual and demo programs.

Only \$49.95! (State Turbo C or Quick C version.)

ONLY
\$49.95
EACH

WINDOWS

Add super-fast text screen handling to your applications with the WINDOWS library from Zortech Inc.

Give your applications the professional look - with instant zooming and exploding windows. Incorporate drop-down menus and Lotus style menus with our easy to use functions.

Automatically handles memory saving and buffering of window text. Use any number of overlapping windows in your applications. Write to any window, read from any window, close any window, pull any window to the top.

Over 55 functions together with a big 85 page manual and remember, you get the full source code.

Only \$49.95! (State Turbo C or Quick C version.)

NEW! C VIDEO

- Now learn C the easy way!
- Get the 'Complete C Video Course' from Zortech Inc. together with our big 365 page workbook.
- Ten 1 hour tapes - 36 lessons!
- Easy to follow course, you get an excellent introduction to the C language.
- Takes you step-by-step up to the intermediate and advanced levels.
- Teach yourself at home or the office - at your own speed.

~~only \$295.00!~~

\$199.95

Yes!
Rush me
these items!

☐ HOTKEY
☐ COMMS
☐ PRO-SCREEN

☐ WINDOWS ☐ GAMES
☐ SUPertext ☐ C VIDEO

FREE SHIPPING - VISA/MC/COD/CHECK

Name

Address

Phone

Exp. Date

VISA or MC#

ZORTECH Inc. 361 Massachusetts Ave, Arlington, MA 02174

ORDER HOTLINE (800) 848-8408

ZORTECH

ASTORIA LONDON FRANKFURT GENEVA

EXAMINING ROOM

Peabody For Turbo C

Product:

Peabody for Turbo C

Target:

IBM PC XT, PC AT, PS/2, and compatibles

Requires:

Hard disk, DOS 2.1 or later, 640K recommended

Pricing:

\$100

Vendor:

Copia International Ltd., 1964 Richton Dr., Wheaton, IL 60187; 312-665-9830

Peabody is one of those programmer's tools that, five minutes after you start exploring it, you wonder how you ever lived without it. A direct competitor to the Norton Guides, Peabody is an online language database for programmers. Peabody also includes some other reference materials and utilities to help developers.

Like the Norton Guides, Peabody comes in various language flavors. The one I used for this review was Turbo C. There are also references for Microsoft and Lattice C, Turbo Pascal 4.0, and DOS.

Peabody is intended chiefly to function as a TSR. You could run it as a standalone application, though I don't know why you would except for familiarization. Additionally, there's a mode called "tandem" in which Peabody becomes a temporary TSR. For tandem mode, you type a command such as *PEABODY TC*. This brings up Peabody in standalone mode, but Peabody starts Turbo C as a child process and then hovers in the background pretending to be a TSR. Similarly, you can run Peabody in tandem with Brief using the command *PEABODY B filename.ext*. An exit from the child also ends Peabody and removes it from memory.

Ron Copeland, associate editor for DDJ, is the coordinator for this review section. He welcomes your feedback on products worth reviewing.

This is an attractive feature, because Peabody takes a lot of memory for a TSR: 115,120 bytes. That's 60 percent more than the Norton Guides consume. On the other hand, Norton is a pure TSR and doesn't have a tandem mode; it's either resident or it's not. Peabody's tandem approach makes more sense, since you probably only want to activate the reference system while actually programming, and any other time the resident software wastes memory that could be used for other things.

The disk space requirements for Peabody and the Norton Guides are about the same, with both taking around 700K. The Peabody reference database is a little bigger: 542K versus 516K for Norton.

Peabody uses a minimum of four hot keys. For the Turbo C version:

- Ctrl-Tab brings up the database table of contents
- Alt-LShift serves as the Hyper-key
- LShift-Tab redisplay the most recent frame
- Ctrl-Backspace tags the current Peabody window as a "sticky frame"

These default hot keys are reassignable with a configuration utility called *PBSETUP*. Each additional Peabody database you install brings along its own default Hyperkey; Turbo Pascal, for example, uses LShift-Ctrl.

Hyperkeys and sticky frames are features that programmers are sure to love (although it sure would be

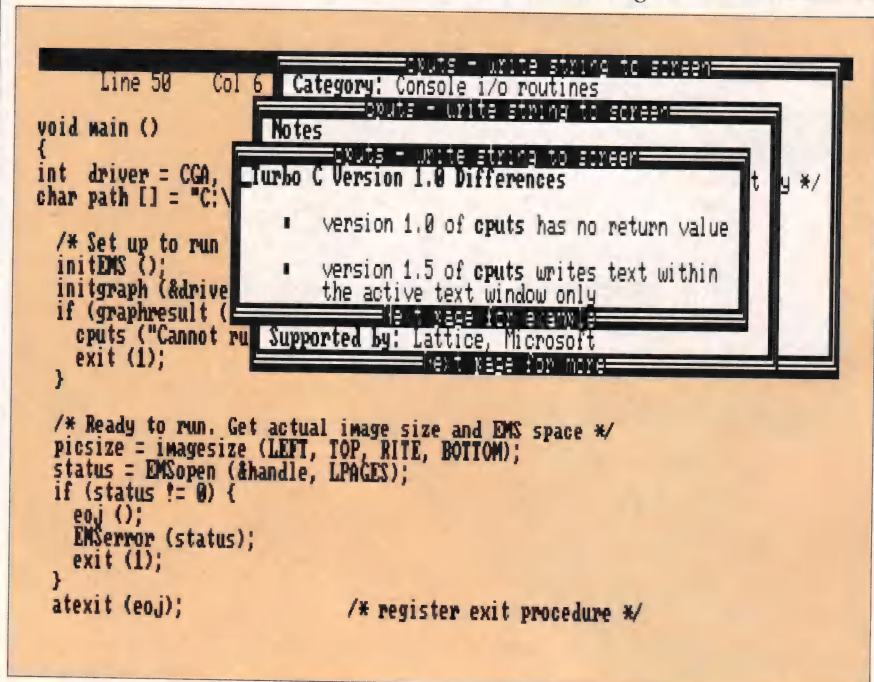


Figure 1: Field Validation Options for PC/Forms

Employment Wanted

Fast worker!
No job too big!
References available:
Call (800) 541-1261

Ask for
Microsoft FORTRAN 4.1

Our FORTRAN is hungry for work. And it can handle any job you've got. Whether you're writing a math-intensive engineering program on your PC or porting a huge scientific application down from a mainframe, Microsoft FORTRAN Optimizing Compiler version 4.1 is the FORTRAN for you.

It comes with an impressive resume and a great track record. And with new support for the OS/2 systems along with standard DOS support, it's a real team player that fits in with any organization.

Put your FORTRAN development on the fastest track yet.

Microsoft FORTRAN 4.1 generates the fastest FORTRAN programs on a personal computer. Our leading-edge optimizing technology, including loop optimizations, automatically improves the quality of the code it generates.

The result is object code so streamlined, so compact, and so efficient you may think your program is running on a mainframe.

Take on the biggest assignments with OS/2 systems support.

Support for the OS/2 systems gives you new capabilities. Write huge FORTRAN applications

that shatter the 640K barrier of DOS. Make direct calls to the operating system. Even create a single "Family API" program that runs under MS OS/2 and MS-DOS.

When it comes to porting huge programs to and from mainframe and mini environments, Microsoft FORTRAN 4.1 can tackle projects of any size—even programs as large as one gigabyte!

And it's still GSA-certified Full and error-free ANSI FORTRAN 77, with the largest set of IBM VS and DEC VAX extensions available for personal computers.

So Microsoft FORTRAN 4.1 makes short work of any porting chore. And you're assured of the highest level of reliability you can get in a FORTRAN compiler.

You also get a great set of tools: the new Microsoft Editor, the famous Microsoft CodeView debugger for both MS OS/2 and DOS, and award-winning documentation.

Give Microsoft FORTRAN 4.1 a long-term contract.

But first check its references for yourself. Just call (800) 541-1261, Dept. E83, for the name of your nearest Microsoft dealer. Then put Microsoft FORTRAN 4.1 to work for you.

Microsoft FORTRAN 4.1



Microsoft FORTRAN

Optimizing Compiler version 4.1

We'll Give You Six Solid Reasons Why You Should Be Developing Your Applications In Clarion.

1 Slash Your Total Development Effort... From Prototyping To Completion

You'll get live results fast—even before you write the first line of code. Ideas become running applications in a few hours.

Designer, the front-end application generator, allows you to produce major programs with *no coding*. It eliminates prototyping as a preliminary step because design and source code generation are done concurrently. You can implement a dazzling color screen or a report in minutes. You'll probably never code a screen again!

2 Automatically Generate Commented Source Code For Your Entire Application

The Professional Developer's **Designer** is the most powerful application generator in the industry. It creates structured source code that is fully commented. You can easily add to it, modify it, or just admire it.

And The Professional Developer is a complete development environment with all the tools that you need.

3 Create High-Speed, Bullet-Proof Data Management With Built-In LAN Support

Advanced techniques allow you to tune your data management to fit each application. Use related files, data encryption, memo fields, automatic recovery, and commit and rollback—all without compromising Clarion's high level of performance. And complete LAN features are included so the applications you develop will run on your network without any additional run-time or LAN PACK cost.

4 Interface To Your Own C or Assembler Routines For Special Requirements

The Professional Developer will support special device routines and complex logic written in C and Assembler. You won't have to completely re-write all those existing procedures. You can produce completely open-ended solutions that can grow with your needs or requirements.

5 Produce Executable Programs That Don't Need Run-Time Systems

Compile your application into an .EXE program that will run on a stand-alone computer or workstation so you can distribute your programs without costly run-time systems.

6 Get Started Immediately. You'll Be Productive The First Day.

Although there's a lot of horsepower "under the hood," you'll be producing programs soon after you open the package. You'll find the whole environment friendly and comfortable. PC Week says: "Clarion is easy to learn and easy to use."



All New Version 2.0

The Clarion Professional Developer Is A Total Programming Environment That Runs On Any IBM PC, PS/2, Or True Compatible With 384K Of Memory And A Hard Disk. The Retail Price Is Just \$695. NOT Copy Protected.

CLARION

PROFESSIONAL DEVELOPER™

Clarion Professional Developer and Clarion Software are trademarks of Clarion Software. Copyright 1988 Clarion Software.

CIRCLE NO. 103 ON READER SERVICE CARD

And One More.

We'll Give You A Free Preview!

See Your Dealer Or Call Toll Free

(800) 354-5444

For A Free Copy Of Our Tutorial Diskette And Introductory Material

(Or, simply return this coupon).

Name

Company

Address

City State Zip

Phone ()

Mail This Coupon To:

Clarion Software

150 East Sample Road, Pompano Beach, FL 33064

swell if Peabody included cut-and-paste, as well). The Hyperkey performs an automatic lookup of the language keyword at the current cursor position. For example, say you've forgotten some of the details of the Turbo C `cputs()` function. You position the cursor on `cputs()` in the source listing, then press Alt LShift, and shazam! Peabody opens a frame explaining `cputs()`.

Norton does the same thing without a special hot key by automatically positioning the expand menu at the keyword indicated by the text cursor.

The Peabody frame is thoughtfully located where it won't overlay the keyword you're worried about. Successive presses of Enter bring up a stack of frames discussing general features, implementation- or version-specific issues, and a short program example. That's what Figure 1, previous page, shows. You can go backward through the stack by pressing Esc and forward again with Enter, removing and adding frames with single keystrokes. Sure beats turning pages in a manual.

A sticky frame is one that remains on screen after you return to edit mode; Norton has no equivalent. When the frame you want to retain is on top of the stack, you press Ctrl-backspace to tag it, then deactivate the Peabody session with Ctrl-Esc. All the Peabody frames except the sticky one disappear. You can move the sticky frame elsewhere with the cursor keys and revert to edit mode with Esc. This is tantamount to leaving an open manual next to the keyboard for further reference as you write the code. Any Peabody hot key evaporates the sticky frame.

The overlapping frames are a mixed blessing. Peabody's use of frames takes less total real estate per unit of information than Norton's quarter- to full-screen panels. That's necessary to implement sticky frames, and it leaves more of your source code visible. On the other hand, the small frames crowd and fragment the information. Norton gives you most or all of the information at a glance in a single, relatively

uncluttered window. Overall, this makes Norton more visually appealing, but the ability to hang on to and move sticky frames as you edit your source code gives Peabody a distinctive advantage.

From the table of contents level (Ctrl-Tab), Peabody furnishes a hierarchy of menus that successively narrow down to the item you want to look up. You can get into the database by subject or keyword, and also by library functions, operators, data types, ASCII characters, and other categories. Norton provides similar paths, but by the alternative means of pull-down menus and cross-references. The outcome is largely the same, but the methods differ. Neither approach seems clearly superior; they're simply different ways of doing the same thing.

Peabody offers a useful utility that lets you examine memory or a file from within the environment in standard dump format. You can also view a directory, which is handy if you're using an editor that doesn't

have a temporary exit to the DOS shell.

The content of the Peabody reference database for Turbo C seems reasonably complete. The only thing that's missing is the graphics subsystem introduced with Turbo C 1.5. This is a curious omission inasmuch as Peabody does furnish information about the text extensions in 1.5. I found no factual errors in the lookup material, whereas I did in Norton; that doesn't mean that Peabody has no mistakes, but just that I didn't notice them if they do exist.

In general, Peabody is an extremely useful, well-rounded programmer's aid that deserves a strong recommendation. For any serious programmer, it will quickly pay for itself in the productivity gains that come from not having to take your hands off the keyboard to look up stuff.

by Kent Porter

(continued on page 128)

UNIX/C WINDOW DEVELOPMENT COMPATIBILITY with CURSES for MS-DOS and MS-OS/2.

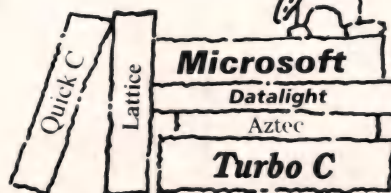
THE BETTER PRODUCT. "Aspen Scientific's Curses library is a fine PC version of System V Curses. Screen updating was fast and clean... has good documentation and is available for many compilers... less expensive... its greater flexibility makes it an attractive package for developers."
Computer Language, June/87

"This is a nice product. If you need Unix-compatible screen output in your programs, or if you just want a nice clean window-management package, I'd recommend it."
Allen Holub, Dr. Dobb's Journal, August/87

Limited Time Offer:
ORDER CURSES NOW
and receive FAST Unix
compatible forms tool
kit with source code
FREE

Complete curses tool kit: **\$119.**
Source code available for: ..\$289.

FORMATION
Window/Menu Option ...\$159.
FORMATION with source ..\$299.



ASPEN SCIENTIFIC

P.O. BOX 72 WHEAT RIDGE,
COLORADO 80034-0072

For technical questions please call
(303) 423-8088

To order NOW please call
1-800-255-5550 ext. 171

C-INDEX +

Product:

C-INDEX +, Version 3.1

Target:

IBM PC, PC AT, PS/2, and compatibles

Requires:

Unix System V, Xenix System V.PC; any operating system that can run Lattice C, Version 3.1; Computer Innovations C86, Version 2.30; Microsoft C 3.0, Version 3.0 or 4.0 including OS/2; Consulair C, Version 4.5

Pricing:

\$395

Vendor:

Trio Systems, 2210 Wilshire Blvd., Ste. 289, Santa Monica, CA 90403; 213-394-0796

If you'd like to cut down the time required to create and implement systems that need sophisticated file-handling techniques while optimizing the amount of code and storage space needed for those systems, you should consider using C-INDEX +

from Trio Systems. C-INDEX + is a full B-TREE data file management library of individually written functions. Designed specifically for C programmers, these functions can be used in any application requiring fast file creation, update, access, and maintenance schemes.

C-INDEX + handles variable length single- or multiple-key file access of any storage file of fixed or variable length records. It also stores the file index information inside the data file itself. This index storage feature cuts down on the number of data files that are open in an application by eliminating index files. File records utilizing the C-INDEX + functions can be retrieved sequentially, randomly, or by record number. Aside from the physical storage limits of your particular computer system, the only limitations in Version 3.1 are that single records can't be more than 10K and files can't be larger than 32 Mbyte. There are no limits on the number or format type of records in any one file, the num-

ber of fields, or the number of files that can be open at any one time.

One of the other interesting features of this system is that no reorganization of files for the removal of deleted records is necessary. All data space is automatically reclaimed by the system as long as you use variable length records. Error handling is very easy to build into your programs because each function supplied in C-INDEX + handles the error value for you. You merely decide what you want to do when an error is found.

Both single-key and multi-key functions are supplied with the system. The single-key functions include: single and multiuser file creation, file open and file close, file buffer control, add records, change records, delete records, search and retrieve records, multiuser semaphore functions, and multiuser entry locking functions.

The multi-key functions include: single and multiuser file creation, file open and file close, add records,

Conquer Time and Space.

Introducing XO-SHELL™ Pop-Up Productivity for Programmers.

No matter what language you program in, XO-SHELL will help you hurdle the barriers to working faster and more efficiently by eliminating programming hassles. Only with RAM-resident XO-SHELL can you:

- DO CROSS-REFERENCING without leaving your editor
- VIEW ANY FILE and TRANSFER ANY SECTION into your editor or to your printer
- VIEW, COPY and ERASE files directly from a SCROLLABLE DIRECTORY DISPLAY
- With a single key stroke RETRIEVE previous DOS commands, then EDIT and REEXECUTE them
- DO SOURCE-LISTING while in your application
- OBTAIN KEY-CODES without a reference and without going through difficult interpretation
- INSERT GRAPHICS CHARACTERS in your source code.

XO-SHELL is for PCs, XTs, ATs, PS/2s, compatibles.



WYTE CORPORATION
701 Concord Avenue
Cambridge, MA 02138

\$49

plus \$5 shipping & handling

Call today toll-free
(800) 635-5011

In MA: (617) 868-7704
Visa, MasterCard

CIRCLE NO. 255 ON READER SERVICE CARD



Break The dBase Barrier

*dBase to C conversion
is now a reality*

Sooner or later you're going to run into the dBase wall. It may come up unexpectedly. Maybe you know it's there. But at some point you are going to need faster run-time, real portability, stronger code refinement, and source code security.

Using dBx to translate your dBase code to C is the perfect way to break the dBase barrier. C is portable, fast, and flexible. C programmers appreciate our commented, clean K&R code. If you are new to C, dBase is a great way to get up to speed. Why not call us today and discuss your individual situation.

dBx - The dBase to C Translator - It's Real



Desktop Ai
303 Linwood Ave.
Fairfield, Ct., 06430

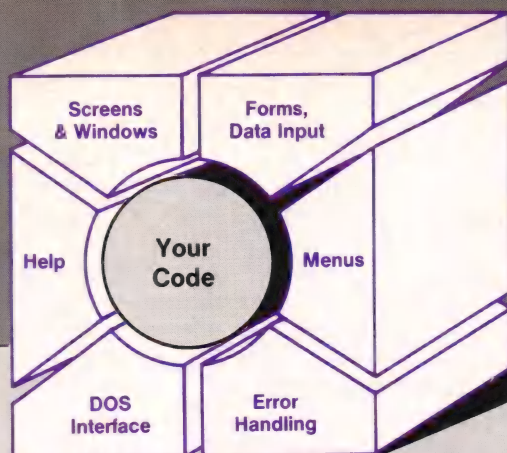
(203) 255-3400

TELEX: 6502972226MCI

CIRCLE NO. 114 ON READER SERVICE CARD

30 day moneyback
satisfaction guarantee

Ask about the
OS/2 version



C-Worthy Interface Library helps you smoothly pull together all aspects of an excellent Human Interface.

C Programmers: Wrap an Exciting, Bullet-Proof Interface Around Your Code Quickly.

Introducing... C-Worthy® Interface Library

The only human interface package you need. That's what our customers are telling us. One early adopter, Novell, Inc. uses it exclusively in the development of their NetWare® Utilities, which reach over 500,000 users. You see, C-Worthy Interface Library is the only library available to handle every aspect of your program's human interface, all in one package. Now your programs will have a consistent look and feel. You no longer have to integrate pieces of libraries from different manufacturers.

As important as you know users are, you often don't have the time to heavily invest in writing routine code. And that's OK, because with over 400 tight, ready-to-use functions, C-Worthy Interface Library takes care of the tedium and lets you spend your time doing what you enjoy. Concentrate on the heart of your application — features that make it unique, special. Let C-Worthy Interface Library do your:

- Menus
- Error Handling
- DOS Interface
- Context Sensitive Help
- Screens, Windows
- Forms, Data Input (optional)

You control color, size, border, location, etc. And if there's anything you want to change, you can. Source is available to provide you with the flexibility you need. And you can distribute your applications freely, with no royalties.

C-Worthy Interface Library requires hard disk media with 256K RAM. MSDOS 2.0+ and IBM PC, or compatible, TI Professional, NEC APC III, or VICTOR 9000. C-Worthy is a registered trademark of Custom Design Systems, Inc.

Tech Specs

- **Compilers:** Microsoft 3.0+, Quick, Turbo, Lattice. All models.
- **350+ functions** written in C, 75+ in Assembler.
- **Menus:** Fully support pop-up, Lotus style, MS Windows style (pull-down), pull-up.
- **Errors:** DOS, program, and user.
- **DOS Interface:** 62 functions. File handling, dir. and drive management, date & time conversion, wildcards, more.
- **Help:** System and context sensitive.
- **Screens:** Screen display, color palettes, save, restore, scroll, more.
- **Windows:** Exploding, tiled, pop-up, overlapping. Direct video access and virtual. Up to 50 active at any time.
- **Keyboard Handling:** Regular, function, interrupt, background procedures.
- **Editing:** String and word wrap text.
- **Form Interface Library:** 118 functions. Over 15 field types, and user definable field types. 3 levels of data validation: type, multiple field ranges, optional validation procedures. Hide, lock, or secure a field. Optimal field movement.
- **Foreign Languages:** All text messages in separate files for easy translation.
- **Compatible with MS Windows.**
- **OS/2** special version when released.
- **Machines:** Autodetect for MDA, CGA, EGA, VGA
- **No royalties.**

"I heartily recommend this package."

— David A. Schmitt, president, Lattice, Inc.

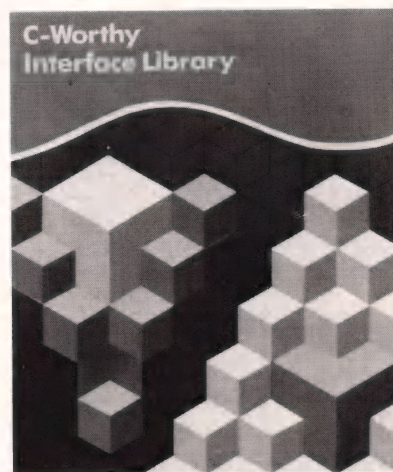
Over 400 developers in 16 countries already use it.

Thorough Documentation

Indexed alphabetically and by category, the 700+ page Reference Guide includes for each function: an example, description, calling conventions, return values, and related functions. The 250 page User's Guide gets you going with its tutorial and "Getting Started" sections.

CIRCLE NO. 229 ON READER SERVICE CARD

Outrageous Demo. Call
for C-Worthy's FREE
Sample Application Kit.
No strings attached.



C-Worthy Interface Library:

Object only	\$ 195
Form Interface Library add-on	\$ 100
Object with Forms	\$ 295
Object with Forms & Library Source	\$ 495

Please specify compiler and version when ordering.

To Order Call

(800) 821- 2492

in MA (617) 337-6963

**Solution
Systems™**

541-L Main Street, Suite 410
South Weymouth, MA 02190

update records, delete records, search and retrieve records, and record locking functions. In the event of any unusual file problems, such as unexpected power outages or disk problems during file reads or writes, Trio Systems has also included a rebuild utility and an index integrity check utility to aid the programmer in the reconstruction of a suspected corrupted file. In this release, local-area networks are also supported with full byte record locking and semaphore (lock flags) functions.

I tested this product in three areas, first to compare functions that I have written and used in my own programs, second for its portability to other compilers, and third to find out what I could do with the least amount of program code and the most amount of C-INDEX+ code. In the first area of the comparison, it was immediately apparent why Trio has one product and has been working to constantly improve it. The

efficiency of the C-INDEX+ code cut my program size down substantially from what it had been when I used my own functions. Converting my existing code to include the C-INDEX+ functions was time consuming because all the function calls had to be restructured or changed. When I wrote a program from scratch, however, the time differential for coding was minimal as I became more familiar with the function parameters required. In addition, when the files were recreated for the newly coded program, the storage space savings was about 25 percent due mainly to the lack of index files. Handling multikey indices and the related record add, change, and delete functions was a far easier task with C-INDEX+.

I chose the Mix C and the ALCOR C systems to test Trio's portability claim because neither of these systems were on the list of directly or indirectly supported C compilers. Since C-INDEX+ is written to very

close K&R C standards, the modification time was minimal. Mix and ALCOR do not include librarian programs, which caused me to take some time making modifications, but I eventually found that this was not a major problem and successfully moved the code to both of those systems. The Trio Systems' claim that their code is portable to other systems is a valid one.

In the third area of my evaluation, the creation of the smallest amount of code, I wrote a small database system with minimal screens and data entry needs in just under 150 lines of code. The approximate breakdown of that code was as follows: 30 lines for variable definition, 30 lines for screens, 40 lines for error handling, 30 lines for C-INDEX+, and 20 lines for miscellaneous. The database stored social security numbers and names and was at best a very small skeleton of a program. I did not need to write any file handling routines, which showed me how many functions are supplied in C-INDEX+. Everything I needed was included in the system.

Although this product is extensively documented, it includes tutorial programs, and requires only calls to the various functions supplied, the user should be familiar with the C language record and pointer structures. In other words, I do not consider this to be a product for the casual C programmer but rather a product that contains a complex and very useful set of library functions for the experienced C programmer.

I heartily recommend this product to programmers who would like to cut down on development time in the data management area of their programs. Record and file handling have always been a problem in the programming arena and C-INDEX+ makes it almost effortless. With the inclusion of source code, the experienced C programmer can usually handle any adaptations to the system that might be necessary in a specific application.

by Neil Freeman

(continued on page 130)

LALR generates parsers for ADA, BASIC, C, Pascal, Modula 2, SQL, dBASE, C++, 386 Assembly, FORTRAN...

LALR 3.0 is a complete LALR(1) parser generator. It's fast, powerful, and easy to use. So, if you're developing a compiler, translator or language interface, you want **LALR**.

If you just want to learn about language translation and state-of-the-art parsing technology, you want **LALR**.

**"unbelievably fast ... can
handle very large grammars"**

COMPUTER LANGUAGE MAG., DEC. 1985



LALR Research • 1892 Burnt Mill • Tustin, CA 92680

* Parser skeletons may be written in other languages.

Requires: DOS 2.0 or later, 256K or 512K for large grammars. Overseas orders: Add \$10.00.

LALR 3.0 includes:

- Grammars for ADA, BASIC, Turbo Pascal and Turbo C.
- Parser skeleton source code with error recovery written in C language.*
- Lexical scanner, syntax checker and calculator source code in C.

60-DAY
MONEY-BACK
GUARANTEE

\$99

714-832-LALR

T-DebugPLUS 4.0

Product:

T-DebugPLUS 4.0

Target:

IBM PC, XT, AT, PS/2 and compatibles

Requires:

DOS 2.0 or later; 256K of free memory; Turbo Pascal 4.0; hard disk, color monitor. Extended/expanded memory recommended

Pricing:

\$45; with source for \$90

Vendor:

Turbo Power Software, 3109 Scotts Valley Dr., Ste. 122, Scotts Valley, CA 95066; 408-438-8608

Like many programmers, I suspect, I tend to prefer PRINT statements for running down bugs. And up until now, there hasn't been a choice with Turbo Pascal 4.0. I'd been striving for days to trap one of those ugly intermittent bugs, just about ready to give up on it, when the new T-Debug 4.0 arrived for review. Five minutes after doing the tutorial, I'd found my bug. It made a believer out of me.

The new T-Debug (despite the official name, this is how the manual refers to it) is a dressed-up version of the earlier Turbo 3.0 debugger. It does for Turbo 4.0 what CodeView does for the Microsoft languages, just as well and just as quickly.

Installation is a painless process that consists of copying a half-dozen files from the delivery diskette, running a utility that patches the two Turbo compilers and TMAP so that they'll support mapping of local variables, and running a setup program for T-Debug. The whole thing takes about a minute.

To prepare programs for the debugger, you compile with mapping turned on (the /\$T+ switch for TPC, or an Options menu selection in the environment). This tells the compiler to produce a map (.TPM) file, which T-Debug uses to find identifiers, symbols, entry points, and so forth. Compiling with the map option doesn't affect the size of the .EXE file.

Unlike the earlier T-Debug for

Turbo 3.0, the new debugger is a standalone program. So if you use the Turbo environment, you'll have

to leave it and run T-Debug separately. T-Debug needs about 250K of memory in addition to the memory

```

352 Var  eqptFlags : word absolute $0040:$0010;
353      videoMode : word absolute $0040:$0049;
354      egaByte   : word absolute $0040:$0087;
355      flag      : word;
356
357 Begin
358   display := ptr ($B800, 0);
359   color := false;
360   flag := (eqptFlags shr 4) and 3;
361   if ((flag = 0) or (flag = 2)) then
362     if videoMode = 3 then
363       if egaByte = 0 then begin           { unless proven otherwise }
C:\TP\OPMSUNIT.PAS                                TDEBUG 4.00
1 $480B:$0114 choice                          $0000 00000000 00000000 0
2 $480B:$034A color                             False
   Watch
Break at : OPMS.PAS\14 $45F3:$0011
* t 2
Break at : OPMSUNIT.PAS\358 $462C:$004D
* w color
* t 3
Break at : OPMSUNIT.PAS\361 $462C:$0072
* e flag
$4947:$1F78 $0002 00000000 00000010 2
*

```

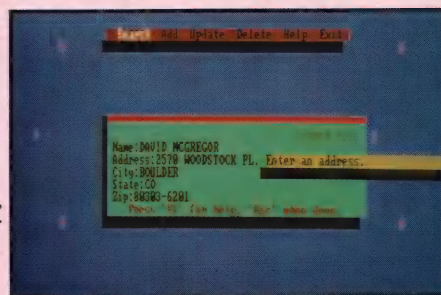
Figure 2: T-Debug divides the display into two basic windows for source and commands.

FREE SOFTWARE

Finally, a Window Library Complete with C Source Code & Make Files!

AEWINDOS works with

- Microsoft 4.0/5.0/QUICKC
- Borland TURBO C
- Lattice C 3.1/3.2



ORDER NOW FOR A FREE 30 DAY NO-RISK TRIAL 1-800-634-5494

Evaluate the package for 30 days under No Obligation. If you like the package (and we're betting you will), keep it and we'll bill you for \$149.95. Otherwise just return the software.

Available now for MS-DOS/PC-DOS, look for MS-OS/2 and Unix versions this summer! Demo disk available for only \$5.00.

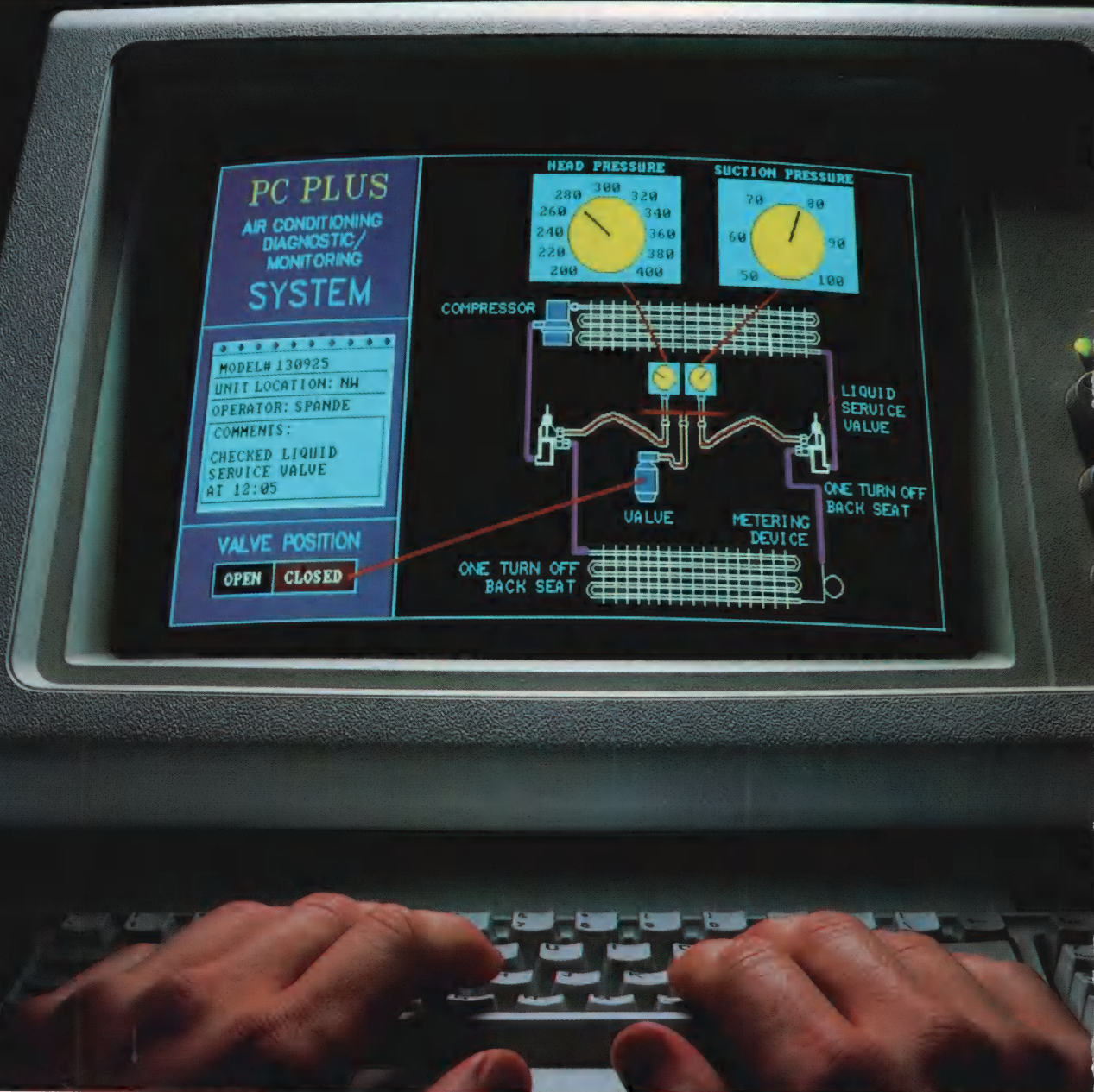
- writes directly to video memory for *SPEED!*
- writes through the BIOS for compatibility!
- comes with a WYSIWYG window editor!
- documentation is TSR, hot-key into it!
- over 100 library functions in K&R C!

**ÆWINDOS
by ÆSOFT**
(303) 499-7332

TURBOC, QUICKC/MA-SOA/MS-OS/2, Lattice C, Unix, and PC-DOS are trademarks of Borland, Microsoft, Lattice, AT&T and IBM.

CIRCLE NO. 77 ON READER SERVICE CARD

Texas Instruments has system developers need.



“Personal Consultant™ Plus... offers a very fine expert system development and delivery tool that already has a proven record with end-users.”

— Susan Shepard, *AI Expert*

Personal Consultant Plus 3.0 Standard Features

- Frames, rules, meta rules and procedures
- Forward/backward chaining
- Confidence factors
- Regression testing and rule tracing
- End-user explanation facilities
- Graphics image capture and display
- Interfaces to dBase™, Lotus 1-2-3™, DOS files, .EXE or .COM programs, "C"
- Complete LISP development environment
- 2-megabyte expanded/extended memory support
- Mouse support
- Context sensitive help
- "Getting Started" tutorial-style manual

Personal Consultant Images

- Optional add-on package to PC Plus (3.0)
- Allows integration of "active images" into

what serious expert Power tools.

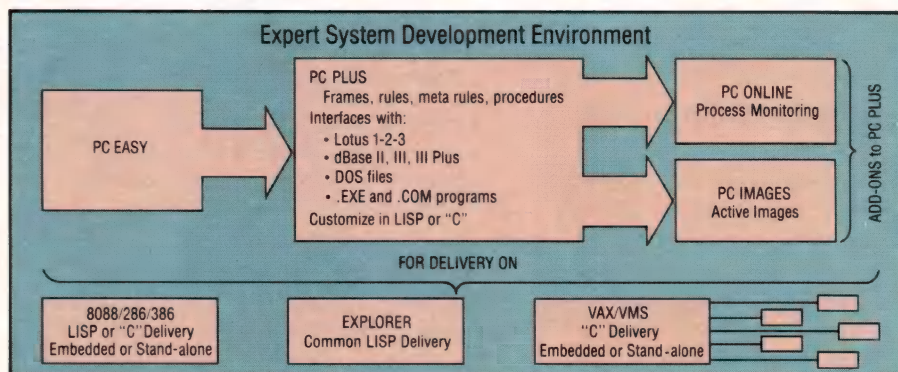
Among all the expert system development tools available for personal computers today, none deliver the power and flexibility of TI's Personal Consultant series.

Personal Consultant Easy is ideal for getting started, and is upwardly compatible with the higher functionality of PC Plus. For experienced developers, Personal Consultant Plus and its optional add-on enhancements, Online and Images, were designed to help solve a broader range of complex problems.

package helps deliver expertise that is "online all the time."

Application delivery as flexible as the tools themselves.

Delivery can be in LISP for flexibility, or "C" for maximum speed and portability. Our "C" options support either stand-alone or "embedded" knowledge bases. Options are available for DOS-based PCs, TI's Explorer, and DEC's VAX™ line of multi-user minis running under VMS™.



Personal Consultant Plus. Full power for an affordable price.

At \$2,950, PC Plus has proven to be one of the richest and most flexible problem-solving tools available for the development of complex knowledge-based systems. Designed to take advantage of today's more powerful 286/386 DOS-based computers, or TI's Explorer™ Symbolic Processing System, the new 3.0 version of PC Plus provides powerful standard features and a continuing growth path with the addition of either PC Images or PC Online, or both.

Personal Consultant Images. Picture an expert system with interactive graphics.

At \$495, PC Images enables developers to create knowledge-based applications that incorporate complex graphical "active images." User-interactive dials, gauges, forms and selection images provide a more exciting visual data input and output style.

Personal Consultant Online. The expert system as part of the process. At \$995, PC Online allows the developer to design expert systems which interact directly with process data, as opposed to input from a human operator. Designed for intelligent process monitoring applications, this optional

"Texas Instruments has done more than any other company to educate people about AI, to popularize it, and to make useful AI tools available at reasonable prices."

— Jim Seymour, *PC Magazine*.

Technical support, training courses and Knowledge Engineering Services are available for the Personal Consultant products. If you have a question about any of our expert system power tools, we have the answer.

Pick up the phone and gain a powerful advantage.

Call 1-800-527-3500 for technical overviews of our products and a PC Plus case histories brochure which details how our power tools are being put to work today.

36106

© 1987 TI

Personal Consultant and Explorer are trademarks of Texas Instruments Incorporated.

dBase is a trademark of Ashton-Tate.

Lotus 1-2-3 is a trademark of Lotus Development Corp.

VAX and VMS are trademarks of Digital Equipment Corporation.

* Available 4Q 1987.

- knowledge bases
- Interactive dials, gauges, forms and selection images
- Multiple images can be combined on same screen
- "Getting Started" tutorial-style manual

Personal Consultant Online

- Optional add-on package for PC Plus (3.0)
- ONLINE expert systems that interact directly with process data
- Multiple interfaces to data acquisition and analysis programs
- Knowledge base synchronization with process data
- Functions for historical and predicted trends
- Special user interface/reporting capabilities
- "Getting Started" tutorial-style manual

**TEXAS
INSTRUMENTS**

We Have Ways To Make Your Computer Talk

C Communications Library Has On-Line Debugger

No Computer Is An Island

As we move closer and closer to the paperless society, more people are looking for ways to move information electronically. Essential Communications is the unsurpassed technique for accomplishing that goal in C.

Like all our products, Essential Communications is a comprehensive, completely debugged professional programming tool.

No Assembly Required

Now communicating with other PC's, mainframes, plotters, digitizers, modems or any device that utilizes the RS 232 port is as easy as turning a crank.

Communications programming requires controlling things down to the bit level, a tedious and dangerous job. Our functions allow you to add professional communications without previous knowledge of communications, without loss of data.

You can do all of this without resorting to assembly language. Something you probably don't want to do, even if you know assembler.

We Interrupt This Advertisement To Bring You A Bulletin

Included in the library is a functional bulletin board system (BBS). This system can teach you a lot about communications, and has formed the core of many BBS systems around the country.

We also include a terminal program. Both programs include the source code. The BBS alone is worth the price of the package.

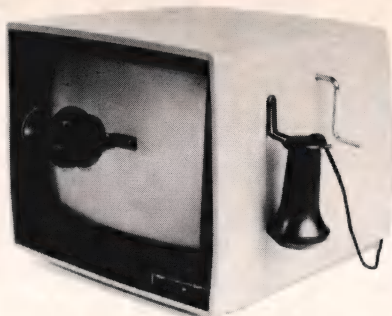
Making The Connection Is Only Half The Problem

Writing communications has its own unique set of problems concerning debugging. Normal debugging techniques are not possible with communications programs.

Is it a software problem? Is it a hardware problem? Am I sending what I think I'm sending? Are the protocols correct? I know, it must be a modem malfunction!

The above products are trademarks of Essential Software.

C Communications



Reach Out And Debug Something

Our combo package includes BreakOut, an interactive on-line data monitor that helps isolate the problems mentioned above.

Giving someone a sophisticated set of communication functions like the Essential Communications Library without an adequate method of debugging just wouldn't be fair.

Power vs Glory

There has always been a trade off in this industry between ease of use and power. Our functions do not require a lot of setups, are well documented and most of all, thoroughly debugged. Essential Communications ease of use stems from our thoughtfulness and not from a lack of power. As with all our products we explain what we are doing every step of the way. Our support staff are all competent C programmers who are thoroughly prepared to assist you after the purchase.

Essential Communications Functions include, general functions, xmodem, timer, keyboard, video, imodem and interrupt services. Please give us a call for information regarding specific features that you require. Source Code Included, 30-Day Money Back Guarantee.

Communications Library	\$185
with BreakOut	\$250
with BreakOut and */resident_c*/	\$350
BreakOut	\$125

Multiple Ports - Up to 8 simultaneously
XMODEM CRC/Checksum/IK
Speeds up to 38.4K Baud
Receive and transmit are interrupt driven
Background communications with
/resident_C/

Multiple XMODEM sessions
Other Essential Products Include:
ScreenStar - Essential Graphics, Utilities



1-800-451-6174
N.J. 201-762-6965
Fax 201-762-0118

Essential Software, Inc.

South Orange Plaza
76 S. Orange Ave., Suite 3
South Orange, N.J., 07079

EXAMINING ROOM (continued from page 131)

requirements of the program you're debugging.

If you're debugging a large application, either extended memory or EMS may be necessary. T-Debug uses whichever is present. If you don't have one or the other, the debugger still works, but it could run out of memory.

Another recommended option is a color monitor. T-Debug makes effective use of colors, as the screen snapshot shows: register values are in red, the next line to execute is highlighted by a blue bar, commands are yellow, and so on.

T-Debug is a command-driven debugger. The lower third of the screen is reserved for dialog, and it scrolls to give a transcript of the last half-dozen interactions. Typical of debuggers, it employs a terse command structure. For example, B sets a breakpoint and -B releases it, G runs the program to the next breakpoint, T traces, and so forth. Many take modifiers: T 5 traces the next five lines of source code, as an example, and G RTN executes the current subroutine up to the point of return. The E command examines a variable or constant. Say you have a variable called COLOR; you can type: E COLOR and T-Debug reports its current value in hex, binary, and decimal, as well as its memory address. The E command is type-sensitive, so Boolean values appear as TRUE or FALSE and characters show up as such.

The command set is quite rich, including such things as the ability to map the heap, determine memory usage, examine the stack, and decompose Pascal source into assembly language. There are also half a dozen commands for defining and managing macros, a powerful feature of the debugger.

Another powerful command set involves watchpoints. If you tell T-DEBUG to watch a variable, it opens a window in the center of the screen. This is just below the register's window in the screen shot. You can watch up to eight variables (12 with EGA and VGA displays). Additionally, you can set conditional breakpoints which automatically halt the pro-

PANEL[®] Plus



Advanced Screen Manager

Building an interactive application in C? **PANEL Plus** provides the features you need for professional program development:

PRODUCTIVITY

The **PANEL Plus** interactive screen design tools are the fastest way to get your application screens set out and tested. Just type prompts on the screen, mark out entry fields, define display and entry attributes, help boxes, borders, pop-up areas. Fields can be edited, moved and resized, and validation details entered – with the screen displayed so that you can see the effect of your changes.

Then **PANEL Plus** saves your work to disk, from where you can either load the design directly into your application program, or for better control, automatically generate C data structures, field areas, and header files which are compiled and linked into your program.

QUALITY

PANEL Plus screens can include all the features demanded by today's applications. Several different menu types are provided, including highlighted bars with help lines. Easy-to-use library functions support pop-up fields, horizontal and vertical scrolling in a field, and validation exits for supplied or custom data checking functions. Text functions can also be carried out in graphics mode using a supported graphics function library.

EASE OF USE

Although the library contains over 150 functions, it is logically organised so that most programs will only need to use a small subset. Documentation is provided with examples of all the main function calls. **PANEL Plus** includes full library source, with variant files for all supported systems, and no royalties are payable for the use of **PANEL Plus** libraries when linked into user applications.

PORTABILITY

PANEL Plus is designed to allow your programs to be ported to just about any environment where you can find a C compiler. Every version of **PANEL Plus** includes source modules for interfacing to: DOS, OS/2 protected mode, Amiga Intuition, Unix (with and without *termcap* or *termio*), Xenix, DOS-J (including 16-bit character editing), and VAX/VMS. Graphics libraries supported include MetaWindow, HALO, Essential Graphics, Microsoft C V5, and Turbo C V1.5. The Microsoft mouse can be used in PC versions.

Now available !

Roundhill announces screen tools for use with the new C compilers from Borland and Microsoft. Special introductory prices:

PANEL/TC – \$129.00

For use with Borland's Turbo C

PANEL/QC – \$129.00

For use with Microsoft's Quick C (after May 1st 1988, \$149.00)

Each package is configured for use on an IBM PC or compatible system, and screens can be designed and built into your programs while running in the special development environment provided with the compiler.

All the **PANEL Plus** library functions are supported, and source code for every validation function is provided so that you can customise the entry checking to suit your application. When you need to move your programs to other environments, or need the full library source for other reasons, upgrades to **PANEL Plus** are available.

PANEL/QC can be run in any of the graphics modes supported by Quick C, and also interfaces to Microsoft C V5. **PANEL/TC** fully supports the Turbo C 1.5 graphics library.

PANEL Plus for MS-DOS or for OS/2, with full library source, is priced at \$495.00. Versions are available for the Aztec, Borland, CI C86PLUS, IBM, Lattice, Mark Williams, MetaWare, and Microsoft compilers. Please call for prices of **PANEL Plus** for Xenix and Unix systems, and for VAX/VMS. Existing registered users of **PANEL** will receive a credit against the **PANEL Plus** license fee.

Roundhill Computer Systems Limited
PO Box 8107, Englewood NJ 07631

(201) 569 2265

Roundhill Computer Systems Limited
PO Box 14 Marlborough SN8 1LR England

(0672) 54675

Telex (UK): 444453 AWARE G
Fax (UK): (0672) 54436

BIX: join roundhill

Roundhill Computer Systems

Unbelievable!

SOURCER™

■ SEE HOW PROGRAMS WORK
■ EASILY MODIFY PROGRAMS

SOURCER™ creates detailed commented source code and listings from memory and executable files. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines necessary assembler directives for reassembly. Complete support for 8088 through 80286, V20/V30, 8087, and 80287 instruction sets. We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER.

On my list of programs that I simply won't do without!

—Robert Hummel, Senior Technical Editor, PC Magazine

SAMPLE OUTPUT

Fully automatic

Program header

Assembler directives

Determines data areas and type

Simulator follows segment changes

Detailed comments

Easy to read format

```
resetprn.lst  ResetPRN v1.01      Sourcer Listing  19-Apr-88  4:05 pm  Page 1
PAGE 60,132
                                RESETPRN
                                Created: 15-Apr-88
                                Version: 1.01
= 0008      data_le      equ      8      ; (0040:0008-378h)
;
seg_a      segment para public
            assume cs:seg_a, ds:seg_a, ss:stack_seg_b
resetprn    proc      far
start:      jmp      short loc_1
            db      'ResetPRN v1.01', 00h
;
            data_2      dw      40h
            data_3      db      00h, 0Ah, 'Reset Printer? $'
;
loc_1:      push      cs
            pop       ds
            mov      dx,offset data_3 ; (658E:0013-00h)
            mov      ah,9
            int      21h ; DOS Services ah=function 09h
;
            mov      ah,1
            int      21h ; DOS Services ah=function 01h
;
            cmp      al,79h
            jne      loc_3 ; Jump if not equal
            mov      ds,data_2 ; (658E:0011-40h)
            mov      dx,ds:data_le ; (0040:0008-378h)
            add      dx,2
            mov      al,8
            out      dx,al ; port 37Ah, printer-2 control
;
            mov      cx,8000h
locloop_2:  loop      locloop_2 ; Loop if cx > 0
            mov      out      al,0Ch ; port 37Ah, printer-2 control
;
            mov      ah,4Ch ; 'L'
            int      21h ; DOS Services ah=function 4Ch
;
            resetprn    seg_a      ends
;
stack_seg_b segment para stack
            db      192 dup (0FFh)
stack_seg_b ends
end start
6593:0000 00C0[FF]
```

(Source code output and inline cross reference can also be selected)

BIOS SOURCE

■ CHANGE AND ADD FEATURES
■ CLARIFY INTERFACES

for PS/2, AT, XT, PC, and Clones

The BIOS Pre-Processor™ with SOURCER provides the first means to obtain accurate legal source listings for any BIOS! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive labels such as "video_mode" and much more. Fully automatic.

SOURCER \$99.95 BIOS Pre-Processor* \$49.95 SOURCER w/BIOS Pre-Processor \$139.95

(Outside USA. Add \$15 Shipping & Handling; CA Residents add local sales tax 6.5 or 7%: *requires SOURCER)

All our products come with a 30 day money back satisfaction guarantee. Not copy protected. To order or receive additional information just call!

800-538-8157 x811
(outside Calif.)

800-672-3470 x811
(inside Calif.)

V COMMUNICATIONS

3031 Tisch Way, Suite 200, Dept. DD, San Jose, CA 95128 (408) 296-4224

PS 2, AT, XT, and PC are trademarks of IBM Corp.

CIRCLE NO. 245 ON READER SERVICE CARD

EXAMINING ROOM

(continued from page 134)

gram if a variable changes or reaches a specified value.

Some of the commands are mapped to function keys, which removes the tiresome need to type a command and press Enter each time you want to execute it. For example, F7 is the same as T (single-step). The F3 key recalls commands from a LIFO stack, displaying them so that you can edit them if necessary and re-execute by pressing Enter.

The only real complaint I have about T-Debug is that it lacks a concise reference to its many commands and function key assignments. The manual isn't terribly large—81 pages including the index—but it's a nuisance to thumb through looking for a specific command. There is on-line help, but you still have to scroll to find what you want. I finally made my own cheat sheet. The vendor should have done it for me.

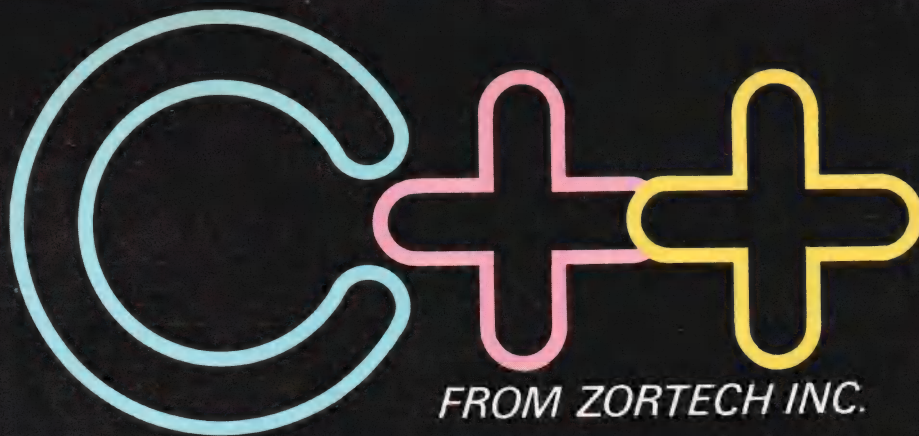
T-Debug will work with dual monitors, which is an important consideration if you do a lot of graphics programming. If you only have one display, T-Debug maintains two screens: its own, and the output of the program. You toggle between them with F10. I had no difficulty tracing a graphics program on the EGA and switching back and forth between text and the drawing. However, there are some gotchas, and the manual devotes nearly four pages to a lucid discussion of them.

On that subject, I encountered a bug when running the EGA in 43-line text mode. Every time I switched from the T-Debug screen to the program's display, the monitor went into 25-line mode and displayed the T-Debug command area in the upper left quadrant. It didn't do any harm, but it was distracting.

T-Debug works fast and well, and it has a well-rounded set of features. If you write software in Turbo Pascal 4.0, you need this debugger.

by Kent Porter

DDJ



The change to a pure language

Now, C programmers can move over to C++ with Zortech C++ – the world's first 'true' C++ compiler for MS-DOS machines.

Zortech C++ is a 'true' compiler and fully conforms to Bjarne Stroustrup's specification as outlined in his book 'The C++ Programming Language'.

Previous implementations of C++ were actually 'translators' – only able to translate C++ source code into C. Of course, this was unacceptable due to the long translating and compiling times.

Now, C++ comes of age with the introduction of the world's first true C++ compiler – from Zortech!

■ Object Oriented Programming

C++ is to C what Modula 2 is to Pascal. C++ brings 'classes' to C, so you can create separate modules that contain their own data and data-related operations. These 'classes' then become new types that can in turn be used to create further modules – this allows you to practically create your own language.

■ ANSI C Superset

You don't have to throw away your existing C programs – C++ is a superset of ANSI C. Now, you can take your Microsoft C or Turbo C compatible programs and easily migrate to C++ to take full advantage of the new C++ features.

■ 'Codeview' Compatible

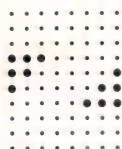
Zortech C++ is compatible with 'Codeview' – Microsoft's industry standard source code debugger.

■ Improved Program Structure

As stated in 'The C++ Programming Language', by using C++ "It would not be unreasonable for a single person to cope with 25,000 lines of code"

■ Other benefits

Here's just a few: Operator overloading, overloading function names, default arguments to functions and better type checking.



ZORTECH

BOSTON LONDON FRANKFURT GENEVA

YES!
Rush me
C++ as shown
below:

☐ Zortech C++ ☐ C++ Book
\$99.95 \$29.95
VISA/MC/COD/CHECK ACCEPTED

Name.....
Address.....
Phone.....

VISA or MC..... Exp. Date.....

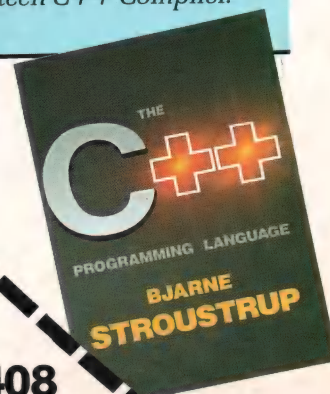
To: ZORTECH INC. 361 Massachusetts Ave., Arlington, MA 02174.
Tel: 617-646-6703. Fax: 617-648-0603.

CALL THE ORDER HOTLINE 1-800-848-8408

CIRCLE NO. 258 ON READER SERVICE CARD

ESSENTIAL READING!

This 325 page book 'The C++ Programming Language' by Bjarne Stroustrup contains the original definition of C++. All the examples shown in this book have been successfully compiled and executed with the Zortech C++ Compiler.





If you want the best there's only

Whether you want the best portable or desktop, the best 80286- or 80386-based personal computer, there is only one choice: Compaq. Because COMPAQ personal computers are consistently rated the best in each class by both industry experts and sophisticated users.

For instance, the COMPAQ DESKPRO 386/20 and the COMPAQ PORTABLE 386 are the most powerful personal computers in the world. Both are based on the 32-bit Intel® 80386 microprocessor, running at a blazing 20 MHz. Both offer the most storage and memory in their classes. And both feature performance enhancements such as concurrent bus architecture, disk caching, and high-speed coprocessor options. All of these features work together to deliver system performance that rivals minicomputers*.

The groundwork for these innovations was laid by the industry's first 80386-based personal computer, the 16-MHz COMPAQ DESKPRO 386. Still outperforming most 80386 machines, it offers high-performance capabilities to users moving up to this class.

In the arena of 80286-based personal computers, the 12-MHz COMPAQ DESKPRO 286 runs your software up to 20% faster than most of its 10-MHz competitors.

No one even comes close to Compaq in portable computing. Because no one but Compaq builds portables with all the features sophisticated users need. The 20-lb. COMPAQ PORTABLE III is the smallest full-function 80286-based computer that truly gives you the power of a desktop. And the COMPAQ PORTABLE II still offers more internal expansion capabilities than any other portable.

*Based on an independent survey of major brands. †Based on an independent survey of 209 FORTUNE 1000 companies.



personal computer, one choice.

Computer users at every level will find that COMPAQ computers represent the best solutions. We've consistently expanded the limits of personal computer technology with advanced features that optimize overall system performance. All while preserving your investment in industry-standard hardware and the world's largest library of business software. Compaq also works to engineer each computer to meet exacting quality and reliability standards, so it's ready to withstand the rigors of the real world.

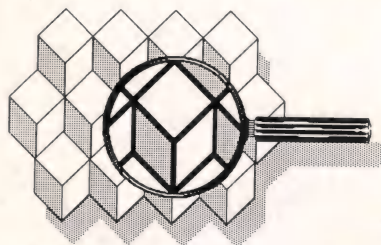
These are all reasons why Compaq earns the highest quality ratings from computer experts. And unsurpassed satisfaction ratings from computer users.* It's also why, this year, more FORTUNE 1000 corporations plan to add Compaq to their approved vendor lists than any other brand.†

If you want the best personal computer, you have only one choice. Call 1-800-231-0900, Operator 50. In Canada, 1-800-263-5868, Operator 50. We'll give you a free brochure or the location of the Authorized COMPAQ Computer Dealer nearest you.

COMPAQ®, COMPAQ PORTABLE II®, COMPAQ DESKPRO 286® and COMPAQ DESKPRO 386® are registered trademarks of Compaq Computer Corporation. ®Registered U.S. Patent and Trademark Office. COMPAQ PORTABLE III™ COMPAQ DESKPRO 386/20™ and COMPAQ PORTABLE 386™ are trademarks of Compaq Computer Corporation. Intel is a registered trademark of Intel Corporation. ©1988 Compaq Computer Corporation. All rights reserved.**COMPAQ**

It simply works better.

OF INTEREST



Language Specific Products

Applied Logic Systems has announced the release of Professional Version 1.2 of the ALS Prolog Compiler for MS-DOS computers. This version is an interactive Prolog compiler designed for programmers building complex intelligent applications and expert systems. Based upon Edinburgh-style syntax, the ALS compiler produces code which executes native reverse at 3,400 LIPS on an IBM PC XT and 31,000 LIPS on a 16 MHz Compaq 386. Features include: a module system, tail recursion optimization, garbage collection, and the ability to create stand-alone .EXE applications. A virtual code space allows programs larger than available memory to run. Price for the compiler is \$499, which includes one year of free updates. Reader Service No. 20.

Applied Logic Systems Inc.
P.O. Box 90
University Station
Syracuse, NY 13210
315-471-3900

Release 5.0 of FORTRIX-C is now available from **Rapitech Systems**. FORTRIX-C is a software converter that provides automated Fortran to C. The new release adds the ability to support all of MIL-STD-1753 Fortran, all but five VMS Fortran enhancements, and virtually all ANSI-Fortran-66. Other new features include: an improved error handler with more complete diagnostics; a 30 percent reduction in the size of the output module; a post-processor that the user may elect to use to

eliminate awkward (though correct) C constructs in the translated code; a makefile generator that creates a dependency script for the Unix system *make* utility; and a simple common handler designed to generate more efficient C source code under certain conditions. Reader Service No. 21.

Rapitech Systems
Montebello Corporate Park
Suffern, NY 10901
914-368-3000

BBx, from **BASIS**, is a derivative of Basic enhanced for business data processing. It offers file structures from flat files to multi-keyed files. Intrinsic locking at the record and file level prevent corruption of data during concurrent access. Designed for interactive processing, BBx provides data verification and error handling allowing development of user-proof applications. Decimal arithmetic with programmer specified precision eliminates undesired rounding of calculations. Device-independent I/O facilitates the use of terminal screen manipulation (including windowing), printer forms control, and graphics devices. All implementations of BBx are binary compatible with each other.

BBx is available for MS-DOS, PC-DOS, Xenix, and Unix on over 65 different computers. All versions of BBx include a full set of developer utilities. Reader Service No. 22.

BASIS Inc.
P.O. Box 20400
Albuquerque, NM 87154
505-821-4407

HiSoft has just released two new language products for the Atari ST: PowerBasic and FTL Modula-2.

PowerBasic is a Basic compiler that includes features such as procedures, functions, local variables, the CASE statement, WHILE and UNTIL loops, and character and integer constants. It also has short and long integer variable types, single- and double-precision floating point numbers and string variables. PowerBasic is a no-limit compiler—no program size limit, no variable size limit, no string size limit, and no array

size limit.

PowerBasic will run on any ST with a disk drive and sells for £39.95.

FTL Modula-2 is a fully standard Modula-2 compiler with linker, 68000 assembler, the complete source code of most of the standard modules, and a host of utilities including a library manager and a complete CLI. Porting source code from other versions is straightforward. Some of the features specific to the ST are a menu creator and a desk accessory builder. The compiler sells for £69.95. Reader Service No. 23.

HiSoft
The Old School
Greenfield, Bedford
United Kingdom
0525-718181

Laboratory Microsystems has announced a new version of the LMI Forth Metacompiler (cross compiler) targeted to Texas Instruments' TMS34010 graphics processor. The LMI Metacompiler runs on an IBM PC, IBM PC AT, IBM PS/2, or compatible with at least 320K and MS-DOS or PC-DOS 2.0 or later. A hard disk is recommended.

The LMI Forth Metacompiler is a professional application development tool. It compiles Forth source code into a stand-alone ROMable or disk-based application. Other features of the compiler include: multi-pass, table-driven compilation; error handling; creation of ROMable or disk based applications; support of local labels and conditional compilation directives; ability to define and invoke new defining words and immediate words in the target code; optional generation of headerless code to conserve memory in the target system; optional compilation from intermediate states; built-in TMS34010 cross-assembler, using standard TI mnemonics; compatibility with the Forth-83 Standard; a detailed 200 page manual; and no royalty or resale licensing fee for well-behaved target applications. The price of the LMI Forth Metacompiler is \$1,000. Reader Service No. 24.

Laboratory Microsystems Inc.
3007 Washington Blvd., Ste. 230



Which would you like to see first? The world's fastest dBASE compiler or the most powerful database development language?

Surprise. Now you get both in the same package.
New Clipper™ from Nantucket®.

Our latest version — Summer '87 — is still the best-performing compiler ever. It lets users run dBASE® applications up to 20 times faster. But there's a lot more to it than raw speed.

Because new Clipper is one of the most powerful, full-featured development languages ever. And gives you more control over your applications than any release of dBASE ever will. Now or in the future.

Instead of designing Clipper as an add-on, we've structured it as an extended database language that uses dBASE as a subset. In addition to emulating the dBASE language, we've added commands for menus, screens, windows and extended functions. As a result, you get dBASE compatibility and an entirely new level of power and versatility.

And with Clipper's open architecture, you can write functions in Clipper, C, Assembler or other languages, and integrate them into one seamless application. Which helps you create more sophisti-

cated applications in less time. And by using our full-featured debugger, you'll be done even faster.

We also give you source code security that keeps users from damaging your application. And sophisticated record and file locking capabilities that make networking applications easier to create. But no matter what you create, you don't have to buy runtime modules or additional software. You don't even have to pay licensing fees.

If you haven't tried Clipper yet, just call (213) 390-7923 today. We'll send you full information and a free demo diskette. Or the complete program, if you'd rather.

But call today. And see how easy it is to find the best dBASE development language. Just get the fastest compiler. And open the box.

Clipper™

Nantucket, 12555 W. Jefferson Boulevard
Los Angeles, CA 90066 Telex: 650-2574125

© Nantucket Corporation, 1988. Nantucket is a registered trademark and Clipper is a trademark of Nantucket Corporation. dBASE is a registered trademark of Ashton-Tate.

OASYS has introduced a family of native and cross development tools

The OASYS Avalon 80386 Assembler/Linker and the OASYS Phar Lap

Borland International is now shipping its Turbo C Run-time Library Source, which offers complete source code to the Turbo C 1.5 library routines, with the exception of the Borland Graphics Interface and math-coprocessor emulation. Run-time Library Source is available to current Turbo C 1.5 owners for \$150. Turbo C runs on the IBM PS/2 and IBM and Compaq families of personal computers and all 100 percent compatibles with 384K, PC-DOS or MS-DOS 2.0 or later, and one floppy drive. Reader Service No. 27.

Borland International
4585 Scotts Valley Dr.
Scotts Valley, CA 95066
408-438-8400

A documentation upgrade is now

• 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 •

$\cdot 386 \cdot 386 \cdot 386 \cdot 386 \cdot 386 \cdot 386 \cdot 386 \cdot 386 \cdot 386$

NOW!

$\xrightarrow[\text{Pascal}]{C}$ **386**

Paradox 386
Foxbase+ 386
386-MATLAB./Weitek

... and others ...

These and other protected-mode 32-bit 80386 programs are among the first to take advantage of the full power of the 386. They and practically every 386 **protected-mode** MS-DOS program that's shipping were done with MetaWare's compilers.

It's no surprise. The recognized leader, MetaWare introduced the *first C and Pascal* compilers that generate protected-mode 386 code for running on any 386 MS-DOS machine (e.g., the Compaq 386 or the IBM PS/2-80) over a year ago. **High C™** and **Professional Pascal™** are well-established and proven.

Smart software developers aren't waiting! Industry leaders such as Borland (ANSA) and Fox use MetaWare's compilers to get dramatic increases in speed and functionality. Don't wait years for Microsoft's 386DOS—your competition will have a big jump on you!

Expand your application to the large 32-bit address space and the full 32-bit registers of the 80386. Go with the long-standing leader. Contact MetaWare for your 80386 software solution today!

(408)429-6382

telex 493-0879



903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060

The Clear Choice for Large Programming Projects —PC Tech Jrm

©1997 Hewlett-Packard Company. All rights reserved. Hewlett-Packard, the Hewlett-Packard logo, and the HP logo are trademarks of Hewlett-Packard Company. Other names and product names may be trademarks of their respective owners.

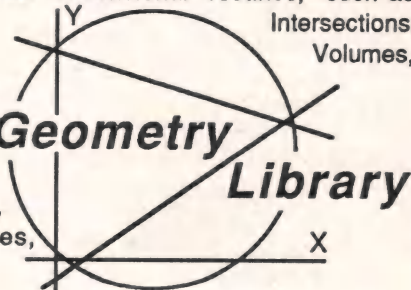
• 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 •

CIRCLE NO. 158 ON READER SERVICE CARD

CAD/CAM, GRAPHICS

Two & Three Dimensional Geometry

The added plus you need for developing sophisticated computer CAD/CAM, graphics & programs that use computational geometry. You save time & money with its flexibility. Over 150 ready to use two & three dimensional routines, such as Arcs, Lines, Polygons, Intersections, Volumes.



TurboGeometry

**Hidden Lines,
Circles, Curves,
Clipping,
Geometric**

Equations, 2 & 3 dimensional Transforms, Areas
& many more... 400 pg Manual, source code and
sample programs. \$99.95US, add \$5.00 for S&H.
in US. TX Res add 8% ST. 30 da guarantee. MC,
VISA, MO, Chk. PC(Comp). Turbo Pascal 4.0 & C,
MS C. MSDOS 2.0+. To order call 214-423-7288

or write to:

Disk Software, Inc.
2116 E. Arapaho, #487, Richardson, TX 75081

CIRCLE NO. 117 ON READER SERVICE CARD



Our front end helps protect your back end.

Today's users require sophisticated interfaces for their applications. Yet complex front ends are a real pain to create, especially when the specs change and your deadlines don't.

But now JYACC introduces JAM™,



Use windows and colors freely with JAM.

a powerful user interface development tool that makes it easier than ever to design and revise

your complex applications.

JAM is the first tool that does it all, from prototyping to implementation. With JAM, you start by creating screens and linking them together to develop an application shell. You can experiment with the look of the interface, and even explore "what if" scenarios on the application flow. Then you attach processing routines, and your application is complete.

JAM works under the following operating systems:

- UNIX®
- MS-DOS®
- VMS®
- XENIX®
- RMX™
- VOS™

You'll be amazed at how quickly your applications spring to life with JAM's interactive screen management utility. You can use features like context-sensitive help, shifting and scrolling fields, a variety of visual attributes and extensive data validations to create exciting screens, windows and menus—all without writing a single line of code. JAM also lets you test your prototypes at any time.

JAM lets you draw from its extensive subroutine library to help you write processing routines faster. And revisions to your applications are easier,

because your subroutines are insulated from the data entry and presentation details of the interface.

JAM is extremely portable. It runs on almost every computer, from PCs to superminis, and works under 6 operating systems. This allows you to develop consistent interfaces throughout your company—a significant asset to the Fortune 500 companies that have been using JAM for more than a year.



Enhance applications with context-sensitive help.

JAM from JYACC. It gets your front ends—and back ends—in great shape. Call for more information about JAM and our demo diskette. **800-458-3313** JYACC FORMAKER™, JAM's screen manager, is also available separately. JYACC, Inc., 116 John Street New York, NY 10038 212-267-7722

Introducing JAM

JYACC Application Manager. *The Composer for Sophisticated Applications.*

JYACC

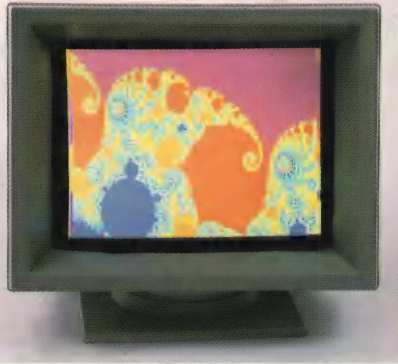
Excellence in Systems & Application Design

MS-DOS, XENIX: Microsoft Corp.; UNIX: AT&T Bell Labs; RMX: Intel Corp.; VMS: Digital Equipment Corp.; VOS: Stratus Corp.

CIRCLE NO. 141 ON READER SERVICE CARD

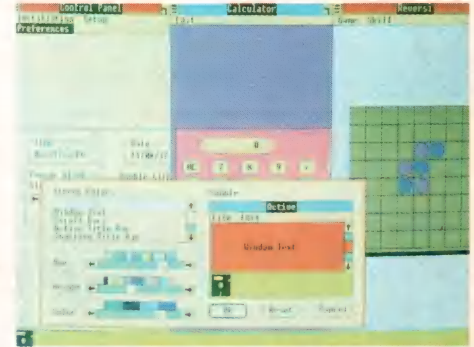
VERY HIGH RESOLUTION

The PC Tech COLOR and MONOCHROME video processor boards employ the TMS 34010 high performance graphics co-processor to insure the best possible video performance at reasonable prices.



Color 34010 Video Processor:

- Featured on the cover of Micro Cornucopia.
- From 800 x 512 through 1024 x 800 resolution (depending on monitor and configuration).
- 8 Bits per pixel for 256 simultaneous colors
- Hardware support for CGA/MDA emulation.
- PC, XT, and AT compatible



The PC Tech Color 34010 video processor is a superior 34010 native code and DGIS development tool. We support up to 4 megabytes of program (non-display) 34010 RAM as well as up to 768K bytes of display RAM. **Compare our architecture and prices to any other intelligent graphics board. Then choose the PC Tech Color 34010 Video Processor for your development engine and your production requirements as well.**

Color 34010 Video Processor\$1,195.00

Price includes 512K display RAM, 1024K program RAM, and utility software. Monitor not included.

Also available: DGIS, 34010 C compiler, assembler, 34010 fractal software, additional display and program memory, and various monitor options.

PC Tech Monochrome 34010 Video Processor and Monitor



- 736 x 1024 resolution (other options available)
- 2 bits per pixel for 4 hardware gray shades
- Hardware support for CGA/MDA/Hercules emulation
- PC, XT, and AT compatible
- Full page 66 line text editing with many popular editors
- Excellent windows 2.0 application development system

The graphics and bit manipulation capabilities of the TMS 34010 make the PC Tech Monochrome 34010 Video Processor 66 line full page text and graphics display faster than many 25 line systems. The video processor is available separately or with the high resolution white phosphor monitor shown above.

Monochrome 34010 Video Sub-System\$1,295.00

Price includes Monochrome Video Processor and monitor pictured above.

Also available: DGIS, TI 34010 C compiler, TI assembler.

Monochrome 34010 Video Processor also available separately.

Designed, Sold and Serviced By:



904 N. 6th St.
Lake City, MN 55041
(612) 345-4555
(612) 345-5514 (FAX)

PC, XT, and AT are trademarks of International Business Machines Corp.

CIRCLE NO. 183 ON READER SERVICE CARD

READER SERVICE

Knowing a little more about you helps us to tailor *Dr. Dobb's Journal* to specific reader interests. Please take a moment to answer the questions below before you return this card for free product information. Thanks very much!

A. Do you design or develop software for professional applications? 1. ☐ Yes 2. ☐ No

B. What type of computer(s) do you use at work? Please check all that apply.

- | | |
|---|---|
| 1. <input type="checkbox"/> IBM PC or XT | 8. <input type="checkbox"/> Macintosh Plus |
| 2. <input type="checkbox"/> IBM AT | 9. <input type="checkbox"/> Macintosh II |
| 3. <input type="checkbox"/> IBM PS/2 (all models) | 10. <input type="checkbox"/> Macintosh SE |
| 4. <input type="checkbox"/> Other IBM | 11. <input type="checkbox"/> Sun |
| 5. <input type="checkbox"/> PC or XT compatible | 12. <input type="checkbox"/> Apollo II |
| 6. <input type="checkbox"/> AT compatible | 13. <input type="checkbox"/> CP/M, TRS-80, or Apple |
| 7. <input type="checkbox"/> 80386 computer | 14. <input type="checkbox"/> Other |

C. Please estimate the total amount you anticipate spending on new hardware and software (through approval, recommendation, or specifying) in the next 12 months:

- | | |
|--|--|
| 1. <input type="checkbox"/> \$1000 or less | 4. <input type="checkbox"/> \$5000 to 10,000 |
| 2. <input type="checkbox"/> \$1000 to 2500 | 5. <input type="checkbox"/> \$10,000 or more |
| 3. <input type="checkbox"/> \$2500 to 5000 | |

D. Does your organization buy hardware from mail order retailers?

1. ☐ Seldom 2. ☐ Occasionally 3. ☐ Frequently

E. Does your organization buy software from mail order retailers?

1. ☐ Seldom 2. ☐ Occasionally 3. ☐ Frequently

Please indicate how often you read the following columns:

- | | | | |
|-----------------------------|------------------------------------|--|-----------------------------------|
| F. Editorial: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| G. Running Light: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| H. Archives: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| I. Letters: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| J. C Chest: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| K. The Forth Column | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| L. To the Macs: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| M. Structured Programming: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| N. Artificial Intelligence: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| O. Of Interest: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| P. Swaine's Flames: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |

Q. What action have you taken as a result of seeing the advertisements in *Dr. Dobb's Journal*?

- | | |
|--|---|
| 1. <input type="checkbox"/> Purchased products | 4. <input type="checkbox"/> Called advertiser |
| 2. <input type="checkbox"/> Recommended purchase | 5. <input type="checkbox"/> No action |
| 3. <input type="checkbox"/> Requested more information | |

R. Please indicate which of the following languages you use professionally:

- | | | | |
|---------------------------------------|-------------------------------------|--|------------------------------------|
| 1. <input type="checkbox"/> Assembler | 6. <input type="checkbox"/> Fortran | 11. <input type="checkbox"/> Forth | 16. <input type="checkbox"/> Other |
| 2. <input type="checkbox"/> C | 7. <input type="checkbox"/> 4GLS | 12. <input type="checkbox"/> ADA | |
| 3. <input type="checkbox"/> BASIC | 8. <input type="checkbox"/> Cobol | 13. <input type="checkbox"/> Modula-2 | |
| 4. <input type="checkbox"/> Pascal | 9. <input type="checkbox"/> LISP | 14. <input type="checkbox"/> APL | |
| 5. <input type="checkbox"/> dBase | 10. <input type="checkbox"/> Prolog | 15. <input type="checkbox"/> Smalltalk | |

S. Which of the following operating systems do you use professionally?

- | | |
|--|---|
| 1. <input type="checkbox"/> MS-DOS or PC-DOS | 4. <input type="checkbox"/> Unix or Xenix |
| 2. <input type="checkbox"/> OS/2 | 5. <input type="checkbox"/> Real-time operating systems |
| 3. <input type="checkbox"/> Macintosh Finder | 6. <input type="checkbox"/> Other |

READER SERVICE

Knowing a little more about you helps us to tailor *Dr. Dobb's Journal* to specific reader interests. Please take a moment to answer the questions below before you return this card for free product information. Thanks very much!

A. Do you design or develop software for professional applications? 1. ☐ Yes 2. ☐ No

B. What type of computer(s) do you use at work? Please check all that apply.

- | | |
|---|---|
| 1. <input type="checkbox"/> IBM PC or XT | 8. <input type="checkbox"/> Macintosh Plus |
| 2. <input type="checkbox"/> IBM AT | 9. <input type="checkbox"/> Macintosh II |
| 3. <input type="checkbox"/> IBM PS/2 (all models) | 10. <input type="checkbox"/> Macintosh SE |
| 4. <input type="checkbox"/> Other IBM | 11. <input type="checkbox"/> Sun |
| 5. <input type="checkbox"/> PC or XT compatible | 12. <input type="checkbox"/> Apollo II |
| 6. <input type="checkbox"/> AT compatible | 13. <input type="checkbox"/> CP/M, TRS-80, or Apple |
| 7. <input type="checkbox"/> 80386 computer | 14. <input type="checkbox"/> Other |

C. Please estimate the total amount you anticipate spending on new hardware and software (through approval, recommendation, or specifying) in the next 12 months:

- | | |
|--|--|
| 1. <input type="checkbox"/> \$1000 or less | 4. <input type="checkbox"/> \$5000 to 10,000 |
| 2. <input type="checkbox"/> \$1000 to 2500 | 5. <input type="checkbox"/> \$10,000 or more |
| 3. <input type="checkbox"/> \$2500 to 5000 | |

D. Does your organization buy hardware from mail order retailers?

1. ☐ Seldom 2. ☐ Occasionally 3. ☐ Frequently

E. Does your organization buy software from mail order retailers?

1. ☐ Seldom 2. ☐ Occasionally 3. ☐ Frequently

Please indicate how often you read the following columns:

- | | | | |
|-----------------------------|------------------------------------|--|-----------------------------------|
| F. Editorial: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| G. Running Light: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| H. Archives: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| I. Letters: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| J. C Chest: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| K. The Forth Column | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| L. To the Macs: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| M. Structured Programming: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| N. Artificial Intelligence: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| O. Of Interest: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |
| P. Swaine's Flames: | 1. <input type="checkbox"/> Always | 2. <input type="checkbox"/> Occasionally | 3. <input type="checkbox"/> Never |

Q. What action have you taken as a result of seeing the advertisements in *Dr. Dobb's Journal*?

- | | |
|--|---|
| 1. <input type="checkbox"/> Purchased products | 4. <input type="checkbox"/> Called advertiser |
| 2. <input type="checkbox"/> Recommended purchase | 5. <input type="checkbox"/> No action |
| 3. <input type="checkbox"/> Requested more information | |

R. Please indicate which of the following languages you use professionally:

- | | | | |
|---------------------------------------|-------------------------------------|--|------------------------------------|
| 1. <input type="checkbox"/> Assembler | 6. <input type="checkbox"/> Fortran | 11. <input type="checkbox"/> Forth | 16. <input type="checkbox"/> Other |
| 2. <input type="checkbox"/> C | 7. <input type="checkbox"/> 4GLS | 12. <input type="checkbox"/> ADA | |
| 3. <input type="checkbox"/> BASIC | 8. <input type="checkbox"/> Cobol | 13. <input type="checkbox"/> Modula-2 | |
| 4. <input type="checkbox"/> Pascal | 9. <input type="checkbox"/> LISP | 14. <input type="checkbox"/> APL | |
| 5. <input type="checkbox"/> dBase | 10. <input type="checkbox"/> Prolog | 15. <input type="checkbox"/> Smalltalk | |

S. Which of the following operating systems do you use professionally?

- | | |
|--|---|
| 1. <input type="checkbox"/> MS-DOS or PC-DOS | 4. <input type="checkbox"/> Unix or Xenix |
| 2. <input type="checkbox"/> OS/2 | 5. <input type="checkbox"/> Real-time operating systems |
| 3. <input type="checkbox"/> Macintosh Finder | 6. <input type="checkbox"/> Other |

... FOR FAST, FREE INFO,

use this card to find out about the products and services listed in this issue. Simply circle the appropriate numbers below.

Name _____

Title _____

Company _____ Phone (_____) _____

Address _____

City/State/Zip _____

1	21	41	61	81	101	121	141	161	181	201	221	241	261	281
2	22	42	62	82	102	122	142	162	182	202	222	242	262	282
3	23	43	63	83	103	123	143	163	183	203	223	243	263	283
4	24	44	64	84	104	124	144	164	184	204	224	244	264	284
5	25	45	65	85	105	125	145	165	185	205	225	245	265	285
6	26	46	66	86	106	126	146	166	186	206	226	246	266	286
7	27	47	67	87	107	127	147	167	187	207	227	247	267	287
8	28	48	68	88	108	128	148	168	188	208	228	248	268	288
9	29	49	69	89	109	129	149	169	189	209	229	249	269	289
10	30	50	70	90	110	130	150	170	190	210	230	250	270	290
11	31	51	71	91	111	131	151	171	191	211	231	251	271	291
12	32	52	72	92	112	132	152	172	192	212	232	252	272	292
13	33	53	73	93	113	133	153	173	193	213	233	253	273	293
14	34	54	74	94	114	134	154	174	194	214	234	254	274	294
15	35	55	75	95	115	135	155	175	195	215	235	255	275	295
16	36	56	76	96	116	136	156	176	196	216	236	256	276	296
17	37	57	77	97	117	137	157	177	197	217	237	257	277	297
18	38	58	78	98	118	138	158	178	198	218	238	258	278	298
19	39	59	79	99	119	139	159	179	199	219	239	259	279	299
20	40	60	80	100	120	140	160	180	200	220	240	260	280	300

June '88: use before September 31, 1988

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

... FOR FAST, FREE INFO,

use this card to find out about the products and services listed in this issue. Simply circle the appropriate numbers below.

Name _____

Title _____

Company _____ Phone (_____) _____

Address _____

City/State/Zip _____

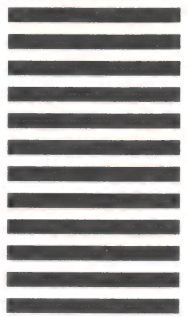
1	21	41	61	81	101	121	141	161	181	201	221	241	261	281
2	22	42	62	82	102	122	142	162	182	202	222	242	262	282
3	23	43	63	83	103	123	143	163	183	203	223	243	263	283
4	24	44	64	84	104	124	144	164	184	204	224	244	264	284
5	25	45	65	85	105	125	145	165	185	205	225	245	265	285
6	26	46	66	86	106	126	146	166	186	206	226	246	266	286
7	27	47	67	87	107	127	147	167	187	207	227	247	267	287
8	28	48	68	88	108	128	148	168	188	208	228	248	268	288
9	29	49	69	89	109	129	149	169	189	209	229	249	269	289
10	30	50	70	90	110	130	150	170	190	210	230	250	270	290
11	31	51	71	91	111	131	151	171	191	211	231	251	271	291
12	32	52	72	92	112	132	152	172	192	212	232	252	272	292
13	33	53	73	93	113	133	153	173	193	213	233	253	273	293
14	34	54	74	94	114	134	154	174	194	214	234	254	274	294
15	35	55	75	95	115	135	155	175	195	215	235	255	275	295
16	36	56	76	96	116	136	156	176	196	216	236	256	276	296
17	37	57	77	97	117	137	157	177	197	217	237	257	277	297
18	38	58	78	98	118	138	158	178	198	218	238	258	278	298
19	39	59	79	99	119	139	159	179	199	219	239	259	279	299
20	40	60	80	100	120	140	160	180	200	220	240	260	280	300

June '88: use before September 31, 1988

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS PERMIT 200 PITTSFIELD, MA

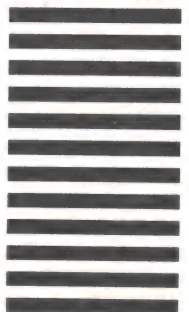
POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

Reader Service Management Dept.
P. O. Box 5282
Pittsfield, MA 01203-9925



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS PERMIT 200 PITTSFIELD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
FOR THE PROFESSIONAL PROGRAMMER

Reader Service Management Dept.
P. O. Box 5282
Pittsfield, MA 01203-9925

The Advertiser Index

Advertiser Name	Page #	RS#	Advertiser Name	Page #	RS#
Aesoft	131	77	Microsoft Press	109	167
AI Architects	47	79	Microware Systems, Corp.	149	163
Aker Corporation	97	*	Migent Software	4-5	164
Alslys	49	80	Mortice Kern Systems, Inc.	86	168
American Cybernetics	147	81	Motorola Corporation	40-41	*
Apollo Computer, Inc.	22-23	*	Nanosoft Associates	61	170
Ashton Tate	54	87	Nantucket Corporation	141	171
Aspen Scientific	127	*	Norton Utilities (The)	36	172
Austin Code Works	84	89	Nu-Mega Technologies	87	174
Basis	16-17	90	Oakland Group, Inc. (The)	120	178
Blaise Computing	2	91	Oasys	43	180
Borland International	1	95	Pathfinder Associates	85	182
Burton Systems Software	77	*	PC Tech	144	183
C Users Group	77	99	Phar Lap Software, Inc.	102	186
CAE/SAR Systems, Inc.	26	101	PIM	101	187
Chen and Associates	95	102	Polytron Corporation	9	191
Clarion Software	126	103	Polytron Corporation	15	190
Coders Source (The)	48	238	Powerline Software Inc.	25	193
Compaq Computer Corporation	138-139	*	Programmer's Connection	150-151	192
Compu View	13	109	Programmer's Market Place	148	*
Courseware Applications, Inc.	71	110	Programmer's Paradise	11	195
Creative Programming	100	*	Programmer's Shop (The)	92-93	197-199
Crosstalk Communications	C4	111	Promula Development Corporation	70	196
Datalight	119	113	QCAD Systems, Inc.	104	200
DDJ Subscriptions	32	*	Quaid Software	33	*
Desktop A.I.	128	114	Quantum Software Systems, Ltd.	37	203
Digitaltalk	44-45	115	Quarterdeck Office Systems	30-31	204
Disk Software	142	117	Raima Corporation	10	194
Dynamic Microprocessor Assoc.	146	118	Raima Corporation	82	*
Ecosoft, Inc.	67	119	Roundhill Computer Systems Ltd.	135	207
Elan Computer Group	66	121	SAS Institute	73	*
Elan Computer Group	113	122	Scientific Endeavors	78	208
Essential Software	136	126	Secom Information Products Co.	105	210
Fagan Microprocessor System (FMS)	46	127	Seidl Computer Engineering	121	211
Gimpel Software	76	*	Semi-Disk Systems	114	212
Golden Bow Systems	58	132	Sequiter Software Inc.	46	213
Golden Solutions	63	134	Sharpe Systems Corporation	103	214
Greenleaf Software	64	136	Soft Advances	78	217
Harvard Softworks	91	137	Softfocus	148	218
Hauppauge Computer Works	27	135	Software Link	C3	219
IET/Inference Engine Technologies	28	138	Software Security, Inc.	99	222
JPI/Jensen & Partners International	21	139	Softway	104	226
JYACC	143	141	Solution Systems	38	228
Kadac Products Ltd.	101	259	Solution Systems	129	229
LALR Research	130	260	Stepstone	108	231
Laboratory Microsystems, Inc.	72	261	Stony Brook Software, Inc.	79	232
Lahey Computer Systems, Inc.	111	262	Summit Information Systems, Inc.	57	233
Lattice, Inc.	59	142	Suncloud Software, Inc.	77	235
Lugaru Software Ltd.	106	144	Systems & Software	122	236
M/SJ Subscriptions	96A	*	Tenon Software	81	*
M&T Catalog of Books & Software Tools	88,96	*	Texas Instruments	132-133	*
Machine Independent Software Corp.	95	146	Tool Makers (The)	81	239
Magna Carta Software	113	148	Turbo Power Software	111	240
Manx Software Systems	7	150	Turbo Tech Report	64A	*
Meridian Software Systems	39	154	V Communications	134	245
Metagraphics Software Corporation	65	156	Vermont Creative Software	62	*
MetaWare Incorporated	142	158	Watcom C for IBM PCs	29	253
Micro Edge	106	160	Whitewater Group	69	254
Micro Way	75	162	Wyte Corporation	128	255
Microsoft	52-53	*	Zortech, Inc.	123	257
Microsoft	115-118	*	Zortech, Inc.	137	258
Microsoft	125	*			
Microsoft Press	107	165			

*The advertiser prefers to be contacted by phone; consult ad.

Advertising Sales Offices

Midwest Charles Shively (415) 366-3600

Northeast Cynthia Zuck (914) 762-9552
Jay Abrams (617) 723-4127

Northern California/Northwest Lisa Boudreau (415) 366-3600

Southern California/AZ/NM/TX Michael Wiener (415) 366-3600

Telemarketing Rep./SE/SW USA Cheri Blum (714) 761-0294

Director of Marketing and Advertising Ferris Ferdon (415) 366-3600

available for BEYOND.BAT, **VM Personal Computing's** package for PC software developers. BEYOND BAT provides PC developers with a script language, panel manager, and full-screen editor so developers can extend the capabilities of the DOS batch language. With the tools included in BEYOND.BAT, developers can front end existing applications, create their own applications, and pre-schedule processing so programs run unattended.

The revisions to the manual enhance its readability and provide a clearer description of the product's three running modes. Illustrations have been added throughout the manual and the index has been extended. BEYOND.BAT retails for \$99. Reader Service No. 28.

VM Personal Computing
41 Kenosia Ave.
Danbury, CT 06810
800-222-VMPC

Miscellaneous Software

IGC has announced that it is shipping version 1.10 of VM/386, The Pro-

fessional MultiTasker. The new version supports the IBM PS/2 and has improved hard disk support, support for VGA, and a shared RAM disk.

VM/386 is a control program that brings true multitasking to users of 80386-based personal computers by combining proven mainframe technology with the advanced architecture of the 386 chip. VM/386 uses the virtual 8086 mode of the 386 to create a series of "guest" virtual machines. VM/386 provides a separate copy of DOS, AUTOEXEC.BAT and CONFIG.SYS for each VM. The VMs run concurrently, and each one runs as if it has all of the resources of the real computer.

VM/386 runs on a wide variety of machines and sells for \$245. Reader Service No. 29.

IGC
4800 Great America Pkwy.
Santa Clara, CA 95054-1221
408-986-8373

Tree86 Version 1.1 is now available from the **Aldridge Company**. Tree86

is a DOS extender and file management utility including unique features such as finding duplicate files, moving subdirectories, built-in mouse support, and EGA-VGA 43-50 row mode. Some enhancements include: the Xcopy command, which gives the user the ability to duplicate directory structures and copy files from one drive to another; the ability to search and work globally on multiple file specifications; the ability to use your own browse/view utility to view files; reporting of total and available expanded memory; and enhanced mouse support.

Tree86 operates on IBM PS/2, IBM PC, IBM PC XT, IBM PC AT, and compatibles with 140K and DOS 2.0 or higher. Suggested retail price is \$49.95. Reader Service No. 30.

The Aldridge Company
2500 CityWest Blvd., Ste. 575
Houston, TX 77042
713-953-1940

DDJ

"HAL"

USE THIS PASSWORD AND SELL YOURSELF ON CHAIRMAN.

If you have a PC and a modem, call (516) 462-6638 and use the password on this page. See for yourself what the first true multiuser bulletin board software can do.

Find out why CHAIRMAN™ is the choice of hundreds of organizations, including *PC Magazine*—for its own popular Interactive Reader Service (up to 2,500 calls a day). *PC Magazine* writes about CHAIRMAN: "For flexibility, speed, ease of setup, security, and company support, you can't go wrong."

SET UP YOUR OWN BULLETIN BOARD

With CHAIRMAN's unique multiuser capability, six callers can simultaneously access a bulletin board on one PC.

CHAIRMAN is the perfect answer for anyone who wants to *distribute* data on a timely basis, *collect* data from numerous sources, or do both. It handles electronic mail

needs at a fraction of the cost of the expensive services, such as MCI Mail,™ The Source,™ and Compuserve.™

Conduct electronic surveys, get questionnaires answered online, send price and delivery information to salespeople, find out support call details without tying up your staff, transfer binary and ASCII files, send software updates to your users, etc.

Combine CHAIRMAN with DMA's FORMULA IV, *PC Magazine* Editor's Choice in DBMS, to use your bulletin board for interactive database applications. CHAIRMAN is available in six-user, two-user, single-user, and network versions. A two-user demo is available for \$25 + \$5 S&H.

For more information, call (212) 687-7115 or write to: DMA, 545 Fifth Avenue, New York, NY, 10017. Ask about our volume discounts.

MCI Mail, The Source, Compuserve are trademarks of their respective companies.
Chairman and FORMULA IV are trademarks of DMA, Inc.

**THE FIRST TRUE MULTIUSER
BULLETIN BOARD**





Multi-Edit vs.

PIZZA

With EVERYTHING!

- Is your editor OUT TO LUNCH?
 - Does it handle ALL OF YOUR NEEDS?
 - Is it flexible, programmable and reconfigurable?
 - MOST IMPORTANTLY, is it EASY TO USE?
- OR WOULD YOU RATHER BE EATING PIZZA?**

Only MULTI-EDIT tastes this good!

Fully automatic Windowing and Virtual Memory

Edit multiple files regardless of physical memory size
Easy cut-and-past between files
View different parts of the same file

Powerful, EASY-TO-READ high-level macro language

Standard language syntax
Full access to ALL Editor functions

Language-specific macros for C, PASCAL, BASIC and MODULA-2

Smart Indenting
Smart brace/parenthesis/block checking
Template editing
More languages on the way

Terrific word-processing features for all your documentation needs

Intelligent word-wrap
Automatic pagination
Full print formatting with justification, bold type, underlining and centering
Flexible line drawing
Even a table of contents generator

Compile within the editor

Automatically positions cursor at errors
Allocates all available memory to compiler

Complete DOS Shell.

Scrollable directory listing
Copy, Delete and Load multiple files with one command
Background file printing

Regular expression search and translate

Condensed Mode display, for easy viewing of your program structure

Pop-up FULL-FUNCTION Programmer's Calculator and ASCII chart

and MOST IMPORTANT,
the BEST USER-INTERFACE ON THE MARKET!
Extensive context-sensitive help
Choice of full menu system or logical function key layout
Function keys are always labeled on screen
(no guessing required!)
Keyboard may be easily reconfigured and re-labeled
Extensive mouse support
Easy, automatic recording and playback of keystrokes
Anchovies easily removed

**MULTI-EDIT COMBINES POWER WITH
EASE OF USE LIKE NO OTHER EDITOR
ON THE MARKET TODAY.**

MULTI-EDIT \$99 COMPLETE

VERSION 2

	Multi-Edit	BRIEF 2.0	Norton Editor	Vedit Plus	PIZZA WITH EVERYTHING
Edit 20+ Files larger than memory	Yes	Yes	No	Yes	12 slices
Powerful high level macro language	Yes	Yes	No	Yes	Italian
Full UNDO	Yes	Yes	No	No	No
Visual marking of blocks	Yes	Yes	Yes	No	Looks Good
Line, stream and column blocks	Yes	Yes	No	No	Use Knife
Automatic file save	Yes	Yes	No	No	No
Online help	Extensive	Limited	Limited	Limited	Extensive
Choice of keystroke commands or menu system	Yes	No	No	Yes	Menu Available
Function Key assignments labeled on screen (may be disabled)	Yes	No	No	No	No
Word processing functions	Extensive	Limited	Limited	Extra Cost	Difficult
Complete DOS shell	Yes	No	No	No	Deep Dish
Pop-up Programmer's Calculator and ASCII Table	Yes	No	No	No ASCII	No
Unlimited 'Off the Cuff' keystroke macros	Yes	No	No	Yes	Sauce on Cuff often
Allocates all available memory to compiler when run from within editor	Yes	No	No	No	Lots of bytes
Intelligent indenting, template editing and brace/parenthesis/block matching and checking for C, PASCAL, BASIC and MODULA-2	Yes	C Only	No	Limited	Limited Intelligence
Flexible condensed mode display	Yes	No	Yes	No	Definitely
PRICE	\$99	\$195	\$50	\$185	About \$12

Get Our FULLY FUNCTIONAL DEMO Copy for only \$10!

To Order, Call 24 hours a day:
1-800-221-9280 Ext. 951
In Arizona: 1-602-968-1945
Credit Card and COD orders accepted.

American Cybernetics
1228 N. Stadem Dr.
Tempe, AZ 85281

Requires IBM/PC/XT/AT/PS2 or full compatible, 256K RAM, PC/MS-DOS 2.0 or later. Multi-Edit and American Cybernetics are trademarks of American Cybernetics. BRIEF is a trademark of Underware, Inc. Norton Editor is a trademark of Peter Norton Computing, Inc. Vedit is a registered trademark of CompuView Products Inc. Copyright 1987 by American Cybernetics.

**How to place your ad in
Programmer's Marketplace:**

Carefully type your message or send camera-ready art. Each ad is 10 x 13 picas (1 5/8 x 2 3/16 inches). All ads must be prepaid. We accept checks, money orders, Visa, Mastercard, and American Express.



Send your ad along with your payment to:
Programmer's Marketplace,
Dr. Dobb's Journal, 501 Galveston Drive,
Redwood City, CA 94063.
For more information, contact:
Glynn Mansfield at (415) 366-3600.

To build a 'DISKLESS' PC or AT use
EDISK (nonvolatile EPROM/RAM disk)

Applications: Turnkey Workstations, Industrial Controllers. **Features:** Self-Booting, EPROM disk programs online, Replaces 1 or 2 floppy disks, Battery backed RAM.

Prices: 180K eprom read-only \$290, eprom p.s. \$50, 720K eprom (R/W) \$580, 360K eprom (R/W) + 180K ram \$580

(604) 852-1155

WIZDOM Computer POBox 121,
Lynden, WA 98264 (or) 1603 Mt. Lehman Rd.
Abbotsford, BC, Can. V2S-5W6

CIRCLE NO. 281 ON READER SERVICE CARD

SOFTWARE/SCIENTIFIC

**ORDINARY/PARTIAL
DIFFERENTIAL EQN
SOLVER**

FOR THE IBM PC & COMPATIBLES

MICROCOMPATIBLES INC.

301 Prelude Dr., Silver Spring, MD 20901

(301) 593-0683

CIRCLE NO. 282 ON READER SERVICE CARD

'C' DOCUMENTATION TOOLS

Save hours of debugging and documentation!
Document the internal structure of 'C' programs.

***C-CALL** (\$39) gives XREF or graphic-trees of all the system/module caller/called structures.

***C-REF** (29) gives variable/constant XREFs. (incl. global vs local) at system or module level

***C-HDR** (\$39) inserts/updates module headers showing caller/called and local/global/params.

***C-LIST** (\$29) lists and/or action-diagrams, also reformats (incl. comment re-alignment).

***SPECIAL !!** All 4 for \$99 (Visa/MC accepted) 30-day money-back guarantee of satisfaction

SOFTWARE BLACKSMITHS INC
6064 St Ives Way
Mississauga, ONT, Canada. L5N 4M1
(416) 858-4466

CIRCLE NO. 283 ON READER SERVICE CARD

SOFTWARE DEVELOPERS!!!

A software company in Scandinavia wants to distribute your easy-to-use mini or micro computer software. Using a dynamic direct sales and marketing data base, we can get your products selling fast. Contact Profesyhtiot, Sinikalliontie 14, 02630 Espoo FINLAND. Phone 358-0-502-1344 or FAX 358-0-502-1504.

CIRCLE NO. 284 ON READER SERVICE CARD

SCREEN MANAGER

P·R·O·F·E·S·S·I·O·N·A·L

• Source code included • Unlimited Number of windows • Complete writing and scrolling support for overlapped windows • EGA, CGA, MGA support • Selectable direct video or BIOS access • Multiple Monitor support • No Fuzz or Flicker • Drag windows around the screen • Easy to use manual • Over 70 Functions • Mult-Tasking • No Royalties

Logical Alternatives, Inc.

CALL (814) 234-8088

CIRCLE NO. 285 ON READER SERVICE CARD

SOFTWARE DEVELOPERS!!!

A software company in Scandinavia wants to distribute your easy-to-use mini or micro computer software. Using a dynamic direct sales and marketing data base, we can get your products selling fast. Contact Profesyhtiot, Sinikalliontie 14, 02630 Espoo FINLAND. Phone 358-0-502-1344 or FAX 358-0-502-1504.

CIRCLE NO. 286 ON READER SERVICE CARD

**C PROGRAMMERS!
THE TOOLS YOU NEED
AT A PRICE YOU'LL LIKE**

BTree

Supports all index file operations. Very quick sequential or random access, duplicate keys, multiple indices, fixed and variable length data records are all supported.

75.00

ISAM

Works on top of BTree to provide a simple, yet powerful application program/file system interface. Complex filesystem manipulation becomes a snap. Provides the power of a database manager with the flexibility of a programming language.

40.00

lp

Finally, a completely device independent printer library! lp drives any printer as accurately as possible and allows easy access to its most sophisticated features. Multiple fonts, multi-column output, complex margin formatting, and much more. Pays for itself the first time it's used.

75.00

Snake

The ultimate 'make' utility. We couldn't find a good one, so we wrote a great one. Has all kinds of powerful features including wild card filename expansion, nested macros, and multiple dependency and rules definitions. Ready to go for MS-DOS; C source is there if you use another operating system.

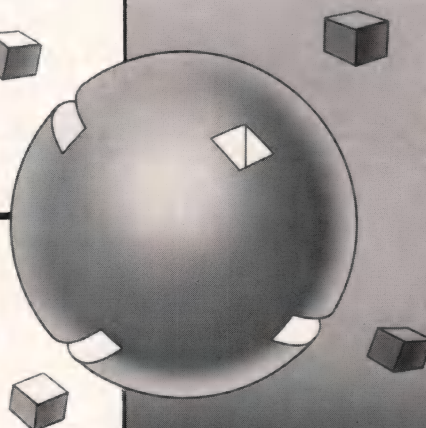
59.00

Combine & Save: BTree + ISAM + lp **159.00** + Snake **199.00**

Each product includes a typeset manual, example programs, and complete C source code that runs on any operating system. Softfocus products may be incorporated into applications royalty-free.

Credit card orders accepted. Visa, M/C, Amex. Dealer inquiries invited.

**NOW
MULTI-
USER
AVAILABLE
60.00**



softfocus

1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

CIRCLE NO. 218 ON READER SERVICE CARD

OS-9...

Making Beautiful Music in the Key of "C"!

When you need to orchestrate beautiful music on your VME system, look to Microware for just the right score. Our finely tuned OS-9 Operating System is truly the maestro's choice when the project requires C Language development. Our superlative C Compiler—also available in an optimized 68020 version—produces fast, compact ROMable code for your most demanding applications. A powerful Assembler, Linker and Symbolic Debugger assists in target development. And our C Compiler is source compatible with UNIX applications and available in cross-compiler configurations for Sun, VAX and Hewlett-Packard environments.

OS-9 Keeps On Performing Even After the Fat Lady Sings!

Most operating systems hit a sour note when the project reaches completion. But not OS-9. Because of its modular design and UNIX-style architecture, your investment in OS-9 experience, tools and applications translates into a valuable resource for your company's future. OS-9 can be utilized again and again over your entire corporate product range.



An Accompaniment of Total Support

Microware is proudly setting the industry's standard for customer support. You'll find outstanding technical documentation that leaves nothing in doubt when it comes to real-world applications. A rigorous Quality Assurance program guarantees customer satisfaction by identifying trouble spots before they become customer problems. And with our Customer Hotline, you are only a telephone call away from courteous and concise information. So join the growing legions of Microware "C" aficionados. Call us today and find out how you can create inspiring harmonies on your system.

microware®

MICROWARE SYSTEMS CORPORATION

1900 N.W. 114th Street
Des Moines, IA 50322
Phone 515-224-1929

Western Regional Office
4401 Great America Parkway
Santa Clara, CA 95054
Phone 408-980-0201

Microware is a registered trademark of Microware Systems Corporation.
OS-9/68000 is a trademark of Microware. VAX is a trademark of DEC.
UNIX is a trademark of AT&T.

CIRCLE NO. 163 ON READER SERVICE CARD

Upgrade Your Technology

We're Programmer's Connection, the leading independent dealer of quality programmer's development tools for IBM personal computers and compatibles. We can help you upgrade your programming technology with some of the best software tools available.

Comprehensive Buyer's Guide. The CONNECTION, our new Buyers Guide, contains prices and up-to-date descriptions of over 700 programmer's development tools by over 250 manufacturers. Each description covers major product features as well as special requirements, version numbers, diskette sizes, and guarantees.

How to Get Your FREE Copy: 1) Use the reader service card provided by this journal; 2) Mail us a card or letter with your name and address; or 3) Call one of our convenient toll free telephone numbers.

If you haven't yet received your copy of the Programmer's Connection Buyer's Guide, act now. Upgrading your programming technology could be one of the wisest and most profitable decisions you'll ever make.

USA 800-336-1166

Canada 800-225-1166
Ohio & Alaska (Collect) 216-494-3781
International 216-494-3781
TELEX 9102406879
FAX 216-494-5260

Business Hours: 8:30 AM to 8:00 PM EST Monday through Friday
Prices, Terms and Conditions are subject to change.
Copyright 1988 Programmer's Connection Incorporated



Established 1984

386 products

	List	Ours
386 ASM/386 LINK Cross Asm by Phar Lap	495	389
386 DEBUG Cross Debugger by Phar Lap	195	145
FoxBASE+ 386 by Fox Software	595	399
MetaWare 386 Products See MetaWare Section	CALL	CALL
Microport 386 Products See Microport Section	CALL	CALL
Microsoft Windows 386	195	129
NDP C-386 by MicroWay	595	529
NDP FORTRAN-386 by MicroWay	595	529
PC-MOS/386 Single-User by The Software Link	195	179
PC-MOS/386 S-Users by The Software Link	595	539
PC-MOS/386 25-Users by The Software Link	995	869
SCO 386 Products See SCO Section	CALL	CALL

american software int'l

DMS Resident-ASM with Source Code	150	139
DMS Resident-C with Source Code	150	139
DMS Screen Master	200	185

assembly language

Cross Assemblers Various by 2500 AD	CALL	CALL
OPTASM by SLR Systems	195	179
risC Assembler Programming Tool from IMSI	80	65

blaise products

ASYNCH MANAGER Specify C or Pascal	175	135
C TOOLS PLUS/5.0	129	99
KeyPilot Super Batch Program	50	47
LIGHT TOOLS for Datalight C	100	59
PASCAL TOOLS/TOOLS 2	175	135
RUNOFF Text Formatter	50	47
Turbo ASYNCH PLUS/4.0	129	99
Turbo C TOOLS	129	99
Turbo POWER SCREEN	129	99
Turbo POWER TOOLS PLUS/4.0	129	99
VIEW MANAGER Specify C or Pascal	275	219

borland products

Paradox 1.1 by Ansa/Borland	495	359
Paradox 2.0 by Ansa/Borland	725	525
Paradox 386 by Ansa/Borland	895	639
Paradox Network Pack by Ansa/Borland	995	725
Quattro: The Professional Spreadsheet	247	179
Sidekick Plus	200	125
Turbo Basic Compiler	100	68
Turbo Basic Database Toolbox	100	68
Turbo Basic Editor Toolbox	100	68
Turbo Basic Telecom Toolbox	100	68
Turbo C Compiler	100	68
Turbo Pascal	100	68
Turbo Pascal Database Toolbox	100	68
Turbo Pascal Developer's Toolkit	395	285
Turbo Pascal Editor Toolbox	100	68
Turbo Pascal Gameworks Toolbox	100	68
Turbo Pascal Graphics Toolbox	100	68
Turbo Pascal Numerical Methods Toolbox	100	68
Turbo Pascal Tutor	70	49

Turbo Prolog Compiler	100	68
Turbo Prolog Toolbox	100	68
Other Borland Products	CALL	CALL

c language

Eco-C88 Modeling Compiler by Ecosoft	100	69
Lattice C Compiler from Lattice	450	289
WATCOM C6.0 by Waterloo Computer	295	269

Peabody Pop-Up Reference Utility by Copia International

Peabody is a fast and flexible on-line reference utility. It provides instant, accurate and complete language information in pop-up frames at the touch of a key. With Peabody, you can select general topics from a structured subject menu, or use Peabody's hyperkey to get instant help for the keyword closest to the cursor.

Peabody Specify 1 Database	100	89
Additional Databases:		
Microsoft C v 5.1 Database	50	45
Microsoft Macro Assembler v 5.1 Database	50	45
MS DOS v 3.3 Database	50	45
Turbo C v 1.5 Database	50	45
Turbo Pascal v 3.0 Database	50	45
Turbo Pascal v 4.0 Database	50	45
Peabody Engine Only	50	45

c utilities

BTree by Softloc	75	69
ISAM File Manager	40	37
C Windows Toolkit by Magna Carta	100	89
C++ by Guidelines	295	259
C-scape by Oakland Group	295	259
C talk by CNS	150	129
CBTREE by Peacock Systems	159	129
CQL by Machine Independent Software	395	329
Curses Window Dev Pkg by Aspen Scientific	119	105
with Source Code	289	249
FOR C by Cobalt Blue	750	679
Graphic by Scientific Endeavors	395	309
Interwork by Block Island Tech	129	115
PC/Forms by Golden Solutions	150	129
QPARSER+ by QCAD Systems	475	CALL
vLIB by Pathfinder Associates	99	89
with Source Code	149	135
WKS LIBRARY by Tenon Software	195	179

creative programming products

Vitamin C by Creative Programming	225	159
Reference Database for Norton Guides	50	47
VC Screen Forms Designer	150	119

database management

Clipper by Nantucket	695	379
dBASE III Plus by Ashton-Tate	695	389

ORDERING INFORMATION

FREE SHIPPING. Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Call for express shipping rates.

NO CREDIT CARD CHARGE. VISA, MasterCard and Discover Card are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include expiration date and authorized signature.

NO COD OR PO FEE. CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of \$100) are available to qualified US accounts only.

NO SALES TAX. Orders outside of Ohio are not charged sales tax. Ohio customers please add 5% Ohio tax or provide proof of tax-exemption.

30-DAY GUARANTEE. Most of our products come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

SOUND ADVICE. Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

INTERNATIONAL ORDERS. Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional \$10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

MAIL ORDERS. Please include your telephone number on all mail orders. Be sure to specify computer, operating system, diskette size, and any applicable compiler or hardware interface(s). Send mail orders to:

**Programmer's Connection
Order Processing Department
7249 Whipple Ave NW
North Canton, OH 44720**

dBx dBASE to C Translator by Desktop AI	550	429
with Source Code	950	729
dFLOW by Wallsoft	149	125
FoxBASE+ by Fox Software	395	249
Geniler by Bytel	395	249
MAGIC PC by AKER	199	179
Paradox See Borland Section	CALL	CALL
Q&A by Symantec	349	299
R-BASE Program Interface by Microm	595	389
R-BASE for DOS by Microm	725	539
for OS/2	895	649
UI Programmer by Wallsoft	295	249

debuggers

Periscope I with Board by Periscope	345	275
Periscope II with NMI Breakout Switch	175	139
Periscope II-X Software only	145	105
Periscope III 8 MHz version	1095	875
Periscope III 10 MHz version	1195	959

digitalk products

Smalltalk/V	100	84
EGA/VGA Color Option	50	45
Goodies Diskette #1	50	45
Goodies Diskette #2	50	45
Goodies Diskette #3	50	45
Smalltalk/Comm	50	45
Smalltalk/V 286	200	175

dos utilities

Desqview from Quarterdeck	130	115
Mace Utilities Paul Mace Software	99	85
XO-SHELL by Wyte Corporation	49	47

elan computer products

Eroff	695	629
NROFF/PC	99	89

essential products

Breakout Debugger	125	89
C Utility Library	185	125
Essential Communications	185	125
Essential Communications with Break Out	250	189
Essential Graphics	299	225
/*resident C*/	99	85
with Source Code	198	148
ScreenStar	99	85
with Library Source Code	198	154

faircom products

c-tree & r-tree Combo	650	519
c-tree ISAM File Manager	395	315
r-tree Report Generator	295	239
d-tree	695	599

gimpel products

C-terp Specify compiler	298	219
for UNIX/XENIX	498	379
PC Lint	139	99
Turbo C-terp for Turbo C	139	119

golden bow products

Vcache	60	55
Vfeature Hard Disk Utility	80	74
Vfeature Deluxe Hard Disk Utility	120	111
Vopt Hard Disk Optimization Utility	60	55

greenleaf products

Greenleaf C Sampler specify QuickC or Turbo C	95	69
Greenleaf Comm Library	185	125
Greenleaf Data Windows Includes Source Code	295	229
for OS/2	395	319
Greenleaf Functions	185	125

jjacc products

JAM JJACC Application Manager	New	750	679
for UNIX/XENIX	New	1350	1189
JYACC FORMAKER	New	495	449
for UNIX/XENIX	New	895	799

C-scape by Oakland Group

List \$279 Ours \$265

C-scape is a powerful and flexible tool for controlling the user interface of C programs consisting of two pieces that may be used individually or together. Look & Feel is a WYSIWYG screen designer and code generator; you may even use Look & Feel to import screens created with Dan Bricklin's Demo Program and turn them into C code automatically. C-scape is a library of functions for data entry, validation, windows, menus, text and help screens. Other features include: pop-up and exploding windows; pull-down, pop-up, and 1-2-3 style menus; built-in error checking; and full color support for CGA and EGA monitors.

komputerwerk products

Finally BASIC Routines	99	85
Finally Modules	99	85
Finally XGraf	99	85

lattice products

Lattice C Compiler for Lattice	New Version	450	289
with Library Source Code		900	495
C-Food Smorgasbord Function Library		150	95
with Source Code		300	179
C-Sprite Source Level Debugger		175	119
Curses Screen Manager		125	85
with Source Code		250	169
dbc III		250	169
with Source Code		500	356
dbc III Plus		750	594
with Source Code		1500	1184
LMK Make Facility		195	138
RPG II Combo All four items below		1400	1099
RPG II Compiler No Royalties		750	625
Screen Design Aid Utility for RPG II		350	309
SEU Source Entry Utility		250	199
Sort/Merge		250	199
Text Management Utilities		120	88

meridian products

Ada Developer Interface	New	170	159
AdaVantage DOD-validated	New Version	795	735
with Optimizer		995	919
AdaVantage Debugger		495	449
AdaVantage DOS Environment Package		50	47
AdaVantage Run-Time Customization Lib	New	2500	2259
AdaVantage Utility Packages		50	47

metagraphics products

MetaWINDOW No Royalties		195	159
MetaWINDOW/PLUS		275	229
QuickWINDOW/C for Microsoft QuickC		95	79
TurboWINDOW/C for Borland Turbo C		95	79
TurboWINDOW/Pascal for Borland Turbo Pascal		95	79

metaware products

High C	New	595	529
386 Version	New	895	799
Professional Pascal	New	595	529
386 Version	New	895	799

microport products

DOSMerge286 Specify 2-Users or Unlimited		249	219
DOSMerge386 2-Users		399	359
DOSMerge386 Unlimited Users		499	439
System V/386 Complete		899	799
386 Runtime System		299	269

386 Software Development System	549	489
Text Preparation System	199	169
System W/AT Complete	649	579
AT Runtime System	249	219
AT Software Development System	299	269
Text Preparation System	199	169

microsoft products

Microsoft C Compiler 5 w/CodeView	New Version	450	299
Microsoft COBOL Compiler with COBOL Tools		700	465
Microsoft FORTRAN Optimizing Comp	New Version	450	299
Microsoft Macro Assembler	New Version	150	105
Microsoft Mouse with Paint & Mouse Menus		150	99
with Microsoft Windows & Paint		200	139
with EasyCAD		175	119
Microsoft OS/2 Programmer's Toolkit	New	350	239
Microsoft Pascal Compiler	New Version	300	199
Microsoft QuickBASIC		99	69
Microsoft QuickC		99	69
Microsoft Windows		99	69
Microsoft Windows 386		195	129
Microsoft Windows Development Kit		500	329
Other Microsoft Products		CALL	CALL

mks products

MKS AWK	75	69
MKS RCS Revision Control System	189	169
MKS SQPS SoftQuad Publishing Software	New	495
MKS Toolkit with MKS Vi Editor	169	129
MKS Trilogy with AWK, CRYPT & Korn Shell	119	105
MKS Vi	75	69

mmc ad system products

C Programmer's Toolbox I	80	69
C Programmer's Toolbox II	80	69
C Programmer's Toolbox Combo	130	115

modula-2 language

LOGITECH Modula-2 Development System	249	199
Modula-2 Compiler Pack	99	79
Modula-2 Toolkit	169	139
LOGITECH Modula-2 Window Pkg	49	39
MODULA-2 by Stony Brook	195	169
with Utilities	345	299

mouse products

LOGIMOUSE BUS with PLUS Pkg by LOGITECH	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
with PLUS & First Publisher	179	139
LOGIMOUSE C7 with PLUS Package	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
with PLUS & First Publisher	179	139
LOGITECH EGA & Mouse	New	399
LOGITECH HIREZ Mouse for High-res Screens	New	149
LOGITECH Series 2 Mouse for IBM PS/2	New	99
Microsoft Mouse See Microsoft Section		

novell development products

Btrieve ISAM Mgr with No Royalties	245	184
Xtrieve Query Utility	245	184
Report Option for Xtrieve	145	99
Btrieve/N for Networks & Multi-user	595	454
Xtrieve/N	595	454
Report Option/N for Xtrieve/N	345	269
XQL	795	599

oregon software products

Pascal-2.....	229	199
SourceTools Make and Version Control.....New	595	539
Network Version.....New	1495	1349

peter norton products

Advanced Norton Utilities	150	8
Norton Commander	75	5
Norton Editor <i>New Version</i>	75	5
Norton Guides <i>Specify Language</i>	100	6
<i>for OS/2</i> <i>New</i>	150	10
Norton Utilities	100	5

phoenix products

Pasm Macro Assembler version 2.0	195	108
Pdisk Hard Disk & Backup Utility	145	99
Plantasy Pac I Phoenix Combo	995	595
Pfinish Execution Profiler	395	209
Phix86plus Symbolic Debugger	395	209
PforCe Specify C Compiler	395	209
PforCe++ Specify C Compiler and C++	395	209
Plink86plus Overlay Linker	495	275
Pmaker Make Utility	125	78
Pmale Macro Text Editor	195	108
Pre-C Lint Utility	295	154

pmi products

EmsStorage	49	45
Graphix	149	119
Macro2	89	79
ModBase	89	79
Repertoire	89	74

Repertoire/Btrieve Toolkit	149	119
----------------------------	-----	-----

polytron products

PolyMake UNIX-like Make Facility	149	135
PolyShell	99	95
PolyXREF Complete Cross Reference Utility	219	199
PVCS Corporate Version Control System	395	359
PVCS Personal	149	135

pop-up reference guides

DocuMotion by NWP-Intelligent Solutions	160	139
Norton Guides Specify Language	100	65
for OS/2	New	150
Peabody by Copia Intl. Specify Language	100	89
Resident Expert by Santa Rita. Specify Lang	CALL	CALL

sco products

XENIX System V for ESDI	New	CALL	CALL
XENIX System V for PS/2		CALL	CALL
XENIX System V 286		1295	979
Development System		595	479
Operating System		595	479
Text Processing Package		195	144
XENIX System V 386		1595	1179
Development System		795	619
Operating System		695	589
Text Processing Package		195	144

software connection products

dB2c FILES	199	179
dB2c TOOLKIT	299	249
dB2c WINDOWS	99	89

software garden products

Dan Bricklin's Demo II	195	179
Dan Bricklin's Demo Program	75	57
Dan Bricklin's Demo Tutorial	50	47

INTERWORK by Block Island Technologies

List \$129 Ours \$115

For XENIX:

List \$150 Ours \$139

INTERWORK is a C program library which allows you to write your programs as a set of concurrent tasks. INTERWORK supports a very large number of tasks limited only by available memory. Low overhead per task results in very fast context switching. INTERWORK is useful for simulation, real-time applications, computer games, and experimentation with parallel programming.

texas instruments

Arborist Decision Tree Software	595	539
PC Scheme Lisp	95	79
Personal Consultant Easy	495	435
Personal Consultant Images	495	435
Personal Consultant On-line	995	869
Personal Consultant Plus	2950	2589
Personal Consultant Runtime	95	84

text editors

Brief & dBrief Combo from Solution Systems	275	CALL
Brief	195	CALL
dBrief Customizes Brief for dBASE III	95	CALL
Epsilon Emacs-like editor by Lugu	195	149
ME Text Editor by Magma Ssrems	New	89 79
with Source Code	New	189 169
Vedit Plus by CompuView	185	128
For XENIX	New	285 229

turbopower products

TDEBUG PLUS 4.0	45	43
with Source Code	90	79
Turbo Analyst 4.0	75	69
Turbo Professional 4.0	99	89

other products

ACTOR by The Whitewater Group	495	439
ApBasic by CompTech Software	100	89
Drawbridge by Courseware, Specify Lang	New	129
HALO '88 by Media Cybernetics	New Version	325
Heap Expander by The Tool Makers		60
Interactive EASYFLOW by Haventree	150	125
MASTER*KEY by Sharpe Systems	80	69
Opt-Tech Sort by Opt-Tech Data Proc	149	99
pcAnywhere by EKD Computer	99	89
Source Print by Powerline Software	97	79
Teamwork/PCSA by Cadre Technologies	995	929
TeleSwitch by EKD Computer	289	259
TLIB Version Control System by Burton	100	89
Tree Diagrammer by Powerline Software	77	65
TurboGeometry by Disk Software	100	89
TurboHALO by IMSI, Specify Turbo C or Pascal	80	75
XenoCopy-PC by Xenosoft	80	75
XenoFont by Xenosoft	50	45

CALL for Products Not Listed Here

SWAINE'S FLAMES

Underlying the hype over HyperCard and CD-ROM there are some pivotal truths, one of these being that well-packaged information is becoming a bigger commodity than ever before. Those software developers and information brokers who can pick up on each others' skills and resources will be able to create a new kind of product, a blend of information and code that goes well beyond what we have seen to date in stackware (better call it heapware).

How these people will create this marvelous new kind of product I do not say because I do not know. However, as an information broker, I must pass along to you software developers this valuable insight into the laws on libel.

The court that heard the appeal in the *Jerry Falwell vs. Hustler* magazine case ruled that *Hustler* magazine had not libeled Falwell because *Hustler's* imputation of sexual impropriety to an evangelist was implausible.

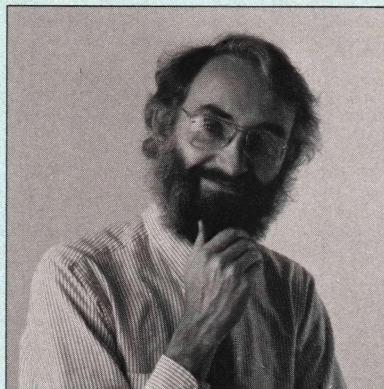
It's the Implausibility Defense, and I see a great future for it. Just make sure your claims are implausible.

But this is all just common sense. You don't need my information broker's insights.

Many truly valuable insights for software developers were being dispensed at Miller Freeman's Software Development '88 this year.

Keynote speaker Jon Bentley outlined his three principles of programming: prototyping, profiling, and the use of little languages.

We heard more about prototyping from Dan Bricklin, who gave some reasons for prototyping—that is, to get nonprogramming experts and potential users involved in the design, to get good products done faster and to weed out bad ideas quickly, and to impress funding sources.



Then, realizing he was preaching to the converted, he gave one typology of prototyping methods (algorithm prototyping, functionality prototyping, appearance prototyping) and some techniques for each type.

Several speakers addressed profiling. Chuck Duff, for one, talked about using the profiling capabilities of Actor to identify the areas of the code that are hogging the processor, so you can selectively change dynamic bindings to static for efficiency.

And William Barrett expanded on the theme of little languages, mentioning that developers working in the Macintosh or VAX VMS or PC AT environment who want the combined functions of Unix *lex* and *yacc* should write to QCAD Systems in San Jose and ask about Qparser+.

And design methodologies. Miller Freeman is big on them. Larry Constantine declared the two powerful principles at the heart of every system design approach to be: Take notes! Draw pictures! Himself a design methodologist, Constantine admitted that modeling a system that doesn't yet exist is a creative endeavor that doesn't yield to techniques that are too structured.

Edward Yourdon was there with copies of his new newsletter, *American Programmer*. It was full of plausible grim prognostications about Japanese and European programmers taking the software industry away from the American programmer. Contact number: 212-769-9460.

Yourdon is no more depressing

than my cousin Corbett, though, who recently told me this tale of software woe.

Immediately after playing Chris Crawford's *Balance of Power*, Corbett decided to write his own game program. It would allow players to choose countries and replay the major wars of the twentieth century. Players could develop nuclear weapons, but if they used them, the program would crash, formatting the hard disk. But the war gaming would be only a tactic in the real strategic goal of the game: global economic dominance.

Midway through the development process, *Esquire* magazine pronounced the death of the Yuppie. It didn't stop Corbett, though. He's hoping that the culture of greed will hang in there long enough for him to make a buck off this product.

It might, but he has now encountered a most frustrating bug. He had gotten as far as developing a testable version of the product and had written an autoplay program to simulate various player strategies when The Bug appeared.

It seems that every time the game is played, the result is the same. The countries that lose the last major war before the atomic age are subsequently prohibited by the victors from developing nuclear arsenals. While the victors invest in nuclear weapons that they will not be able to use without destroying the system, the losers concentrate on education and usable technologies, and achieve global economic dominance in the next round. Same result every time. Boring.

Corbett sees the trap quite clearly. He just doesn't see any way out.

Michael Swaine

Michael Swaine
editor-at-large

Imagine the speed and power of a \$100,000 minicomputer in a desktop PC costing under \$7,000. Now imagine all that power going to waste because the operating system you chose was never meant to take advantage of a computer this powerful. It will take more than just a "window environment" or an outdated operating system to unlock the 80386.

It will take PC-MOS/386™.

The First 80386 Operating System. Specifically designed for the 80386 computer, PC-MOS/386™ opens doors. Doors to more memory and multi-tasking. Doors to thousands of DOS programs as well as upcoming 80386-specific software. It's the gateway to the latest technology..., and your networking future.

Memory Management Without Boards. PC-MOS exploits the memory management capabilities built into the 80386. So, up to four GIGABYTES of memory are accessible to multiple users and to future 80386-specific applications requiring megabytes of memory.

Multi-Tasking, Multi-User Support for One, Five or 25 Users. PC-MOS/386™ allows up to 25 inexpensive terminals to be driven by a single 80386 machine. So the features of the 80386 can be utilized at every terminal. And it comes in three versions so you can upgrade your system as your company grows...without having to learn new commands or install new hardware.

**UP TO
25 USERS.**

**MADE FOR
THE 80386.**

**RUNS DOS
PROGRAMS.**

MULTI-TASKING



Software Support for Thousands of DOS Programs. Although PC-MOS/386™ totally replaces DOS, it doesn't make you replace your favorite DOS programs. So you can run programs like Lotus 1-2-3, WordStar, dBASE III, and WordPerfect on the 80386. Best of all, it uses familiar commands like DIR and COPY—so you'll feel comfortable with our system.

The Gateway to Endless Features. Distinctive characteristics like file/system security, remote access, file/record locking, and built-in color graphics support for EACH user set PC-MOS/386™ apart from all previous operating systems.

Open the Doors to Your Future TODAY! Call The Software Link TODAY for more information and the authorized dealer nearest you. PC-MOS/386™ comes in single, five & 25-user versions starting at \$195.

PC-MOS/386™
MODULAR OPERATING SYSTEM



THE SOFTWARE LINK
Developers of LANLink™ & MultiLink™ Advanced

3577 Parkway Lane, Atlanta, GA 30092
Telex 4996147 SWLINK
FAX 404/263-6474

For the dealer nearest you,
CALL: 800/451-LINK
In Georgia: 404/448-LINK

OEM/Int'l Sales: 404/263-1006
Resellers/VARS: 404/448-5465

OEM/Dealer Inquiries Invited

THE SOFTWARE LINK/CANADA CALL: 800/387-0453

More Than Just Windows, We've Opened Doors.

TRADEMARK ACKNOWLEDGEMENTS: MultiLink® is a registered trademark of The Software Link. PC-MOS/386™ MultiLink® Advanced, and LANLink™ are trademarks of The Software Link. Lotus 1-2-3, WordStar, dBASE III, & WordPerfect are trademarks of Lotus Development Corp., MicroPro, Ashton-Tate, & WordPerfect Corp., respectively. Prices and technical specifications subject to change.
CIRCLE NO. 219 ON READER SERVICE CARD

IBM
Spoken
Here

and
here

and
here

and
here

**Whatever dialect of IBM you need to speak,
CROSSTALK® Mk. 4 makes the connection.**

Now, one program does the job that used to require several. CROSSTALK® Mk. 4 allows high-speed direct communications between PCs and minicomputers, or (with an IRMA™ board) between your PC and an IBM Mainframe, or (with Smart Alec™) between your PC and IBM System 3x's. If you like, CROSSTALK can support all of these sessions (and others) simultaneously, and display each session in its own window.

CROSSTALK Mk. 4 emulates all the terminals you're likely to find useful. That includes IBM 3101 (page and character modes), IBM 525x, IBM 529x, IBM 327x, as well as many popular async terminals like the DEC VT100 and VT220 series. CROSSTALK Mk. 4 includes the powerful CASL™ programming language, which allows you to automate communications applications quickly and easily.

So if you're used to thinking of CROSSTALK just to use with a modem, you're missing some important connections. Ask your dealer for details, or write:

dca® Digital Communications Associates, Inc.
1000 Holcomb Woods Parkway / Roswell, Georgia 30076
1-800-241-6393

CROSSTALK®
COMMUNICATIONS

CROSSTALK and DCA are registered trademarks of Digital Communications Associates, Inc. IRMA, Smart Alec and CASL are trademarks of Digital Communications Associates, Inc. IBM is a registered trademark of International Business Machines Corp. DEC is a registered trademark of Digital Equipment Corp.

CIRCLE NO. 111 ON READER SERVICE CARD

